COMPUTING AND EDUCATION

The Second Frontier

ROBERT O. MCCLINTOCK
Editor

Teachers College, Columbia University
New York and London

Published by Teachers College Press, 1234 Amsterdam Avenue, New York, NY 10027

Originally published in *Teachers College Record*, v. 89, no. 3, Spring 1988.

Library of Congress Cataloging-in-Publication Data

Computing and education : the second frontier / Robert O. McClintock, editor
p. cm. -- (Special issues from the Teachers College record)

"Originally published in Teachers College Record, v. 89, no. 3, Spring 1988"--T.p. verso.
Includes bibliographies and index.

Contents: Introduction : marking the second frontier / Robert O. McClintock -- Image learning : higher education and interactive video disc / Ben Davis -- Moving computing and education beyond rhetoric / Robert P. Taylor, Nancy Cunniff -- Interactive video systems / Carla Seal-Wanner -- Developing thinking skills with computers / John B. Black, Karen Swan, Daniel L. Schwartz -- Computers as material : messing about with time / George Franz, Seymour Papert -- Should computers know what you can do with them? / Don Nix -- Will there be teachers in the classroom of the future?--but we don't think about that / Seth Chaiklin, Matthew W. Lewis.

ISBN 0-8077-2943-4 (pbk.)

1. Education--United States--Data processing. 2. Computer-assisted instruction--United States. 3. Interactive video--United States. I. McClintock, Robert O., 1939- . II. Series.

LB1028.43.C649 1988 88-18577
371.3'9445--dc 19 CIP

Manufactured in the United States of America

93 92 91 90 89 88 1 2 3 4 5 6

Contents

Introduction: Marking the Second Frontier

ROBERT O. McCLINTOCK, *Editor*
Teachers College, Columbia University

Educators have long been responsive to short-lived fads. For a few years this or that seems to hold the key to solving whatever is problematic in education. Thus programmed instruction, competency-based training, back-to-basics, and numerous other causes have caught the imaginations of educators for several years, only to recede into obscurity. Interest in computers and education had such faddish qualities a few years ago, as the microcomputer caught the imagination of many parents, educators, and children. Computing was hailed as the new frontier in education. About 1984, that faddish interest distinctly weakened as people realized that computers in education were no quick fix to educational needs.

One then began to hear the question put with increasing frequency: Would computers in education be another in the series of technological disappointments suffered by educators? Film, radio, television, have all been heralded as revolutionary reforming forces in the realm of education. Despite the ballyhoo, however, they have amounted to developments far less influential than their heralds had predicted. Radio in education is almost a nonexistent influence, and, somewhat more happily, at least, film and television have become useful supplements to established practices, but not anything transformative. Currently computers seem destined to a similar status, patient drill-masters that occasionally supplement didactic explanations with a memorable simulation.

Before acquiescing to this appearance, however, let us reflect on the multiplicity of time scales that are relevant to the flow of human experience. The first interest in computers in education arose with the early excitement over the microcomputer, which took hold as a novelty. That new-frontier status was perforce short-lived, lasting about five years—from 1979 to 1984. The novelty of a technology differs fundamentally from its production duration, however, and we need to find the time scale appropriate for measuring the probable duration of computers as a developing educational influence. This will not be short.

Geologic time unfolds in movements relative to which our lives are mere instants. In contrast, journalistic time consists of units that endure a day or week. The time scale of fashion consists of seasons and years. Politics seems to unfold to rhythms that are slightly longer, say six months to several years, marking the ascendancy of one or another administration. Economics changes at a pace measured in still longer units, very roughly from one year to a decade or more, several if one attends to the longer periodicities of the business cycle. What sort of time scale is appropriate for charting educational change and the role of new technologies in it? It is not the interval according to which pedagogical fads follow one another.

In thinking about the time scale appropriate for marking educational change, one must note when the major pedagogical changes that still condition educational activity began. The current structure of higher education derives largely from the late nineteenth century, when the elective system of undergraduate work and the spectrum of professional schooling were introduced widely. These curricular concerns still seem to be live issues, ones over which people work and disagree, ones capable of yet further development. For instance, a wise translator, Alan Bloom, has now become a best-selling writer by sharply attacking the relativism inherent in this century-old educational structure. Neither Bloom nor anyone else, however, seems to be able to conceive a workable alternative structure to this broad, latitudinarian curriculum of higher education, an alternative that will institutionalize a more ethically rigorous, aesthetically discriminating cultural apprenticeship. Continued criticism of a long-enduring status quo simply indicates that the real period of educational change in this area is well over a century in duration. Compulsory schooling at state expense is also a nineteenth-century idea, on the implementation of which we are still at work, with the prevention of dropouts being an issue of national concern. The twentieth-century idea—lifelong learning or *l'éducation permanent*—is still largely just an idea, with the conditions requisite for its implementation only beginning to emerge in the more advanced communities around the world. Here the appropriate period of educational change would seem not much shorter than the century.

With respect to technical systems, a similar slow, historic pace can be charted. The technological change that is here of interest is not the pace by which design improvements within a given technology may be introduced. These flow into practice every year or so, according to the complexity of the product and the potential profitability of its incremental improvement. The change of interest here concerns the introduction and development to maturity of a major technical system itself. Phone systems, introduced in the late nineteenth century, are still significantly evolving. The Eiffel Tower, representing new architectural technologies of iron and steel structures, elevators, sheath walls of glass and other materials, and the like, will soon be one hundred years old, with the architectural types that it symbolizes still showing

great room for further development. Automobile transportation is a technical system introduced nearly a century ago, and it can be expected to continue evolving for many years to come with major improvements to the safety and economy of the complete system still feasible. Air transport is well over eighty years old; radio is nearly the same; television has existed for over half a century. All still have vast room for further technical development. Such facts about common technologies suggest that technical systems of substantial complexity require one to two hundred years for the development of their full potentialities, perhaps even longer.

As a technical system, computing, like television, is about fifty years old. Most observers would hold, however, that television is a more mature technology, one that has more fully disclosed its potentialities, compared with computing. The reason for this relative maturity is quite simple: The technology of which television is an instance is in fact considerably older than computing. Television is a major subsystem within analog electronic broadcast technologies, which are about eighty years old, whereas "computers" stand for digital electronic systems, a more fundamental, newer technological system, capable of a much broader range of implementation. Analog technologies use changes in one medium, say electromagnetic waves, to represent changes in another medium, say sound waves or changes of illumination along a path back and forth, filling a phosphorescent screen. The system is inexpensive and efficient, but inherently prone to error, which we experience as noise, static, interference.

Digital technologies do not transmit one thing that is analogous to another, the real matter in question. Rather, a digital technology transmits exact, or nearly exact, values, as precisely as these can be represented in binary code. If the real matter in question comprises a set of discrete components, say the letters and words of a text, the members of the set are transmitted as such in binary form. If the real matter in question is, in contrast, a continuous-wave phenomenon, a representation of the wave is created, consisting of numerous samplings of its values at discrete intervals, and the values of these samplings are transmitted in a way that is inherently resistant to error, for the code is direct and simple and subject to error detecting and correction. The key to digital technology, compared with analog, is the digital absence of ambiguity: It deals with successive states, either-or conditions in which a circuit is either off or it is on. In contrast, the analog technology deals endlessly with the torturing indefinite in which each successive state differs from its predecessor by a nearly infinitesimal increment. The analog approximates one whole with another; the digital samples the whole and reconstructs it from that sampling.

In this sense, digital technology is a radical innovation. Insofar as something can be described accurately in binary code, it can be *recreated* from that code. The matter is not merely approximated or represented, but fundamen-

tally subject to recreation, a second instance of the thing, not a mere copy. This condition is one of the technological sources of the more recalcitrant human problems of spreading computing technologies. For instance, copyright laws seem to break down in digital environments because the familiar dynamics of reproduction do not seem to hold. Copies, in the familiar analog realm, are costly to make and at best approximate, leaving clear traces of what is the original and what is the copy. In the digital realm, copies are nearly costless, they are often indistinguishable from the original, assuming some real meaning to "original" can in fact be attached to something substantial. In short, digital computers using binary code to describe and act on all manner of things are a profoundly new technology, one that will probably have, relative to other modern technologies, a very, very long period of development. Thus the significance of computing and its educational influence should be measured along a duration of at least one or two centuries, if not considerably more, and of that duration, only fifty years have passed.

Recognizing now that the educational significance of computing should be measured on a long time scale, we are still left with the question of what that significance is. Computers are artifacts, designed and manufactured tools, whereas education is a preeminently cultural phenomenon, something that takes place through and for people. The history of education is not coextensive with the history of educational tools and stratagems. Will the cultural consequences of computers be contained *within* the culture, facilitating familiar activities within it without overtly changing human repertories of thought and action? Or will those artifacts have a substantial influence *on* the culture, empowering people to act and think in ways in which they could not have formerly acted or thought? To come to grips with this question we need to reflect, not on the technological evolution of computing, the first frontier, but on the evolution of its cultural influence, the second frontier.

Unfortunately, discussions of technology and culture often proceed with an ill-defined set of conceptual distinctions. In particular, commentators chronically render questions of determinism simplistic because they address the phenomena with inadequate concepts. In what follows, I shall suggest that we are in the midst of historically irreversible change, and with that some readers will be inclined to dismiss the suggestion as an example of technological determinism. It is not. Numerous phenomena are voluntary and nondeterministic, but irreversible once commenced. When irreversible actions have been initiated, they must be allowed to carry through to their conclusion, perhaps with some adjustment concerning the duration of the process and the pattern of attention associated with it, but without much voluntary control over the unfolding process. Swallowing is a good example. To swallow or not to swallow is a voluntary behavior. But having started to swallow something, I cannot casually decide midway through the action to unswallow it, or even to stop the swallow except with the risk of results that will

certainly be undignified if not dire. History writing would be much more illuminating than it often is if more attention were paid to understanding the dynamics of irreversible actions without the premature brickbats of deterministic explanations being bandied forth.

Irreversible actions are actions the initiation of which is entirely contingent, but that once initiated follow an inherent course that has a set direction in time. Will it as you may, you do not grow younger. The boy at the edge of a stream may ponder for many minutes whether to leap or not, but once he coils and leaps, he cannot unleap, but must let the action carry through to a splash or a dry landing, whichever his strength, judgment, and the actualities decree. Most action is irreversible in this sense. I hope that my house will not catch fire, but should it catch fire and start to burn, I cannot reverse that phenomenon; I can at best put the fire out and repair the damage. Yes, we do try to intervene in irreversible phenomena and force them to follow a more desirable course than they would follow without the intervention. Note, however, the conceptual structure of such intervention: We recognize a normal course for the phenomenon—the materials of the house and the laws of combustion. Then we try to devise strategies for altering the normal course by adding further causalities to those irreversibly at work.

In its broadest sense, a *frontier* is what is crossed when an irreversible action has been initiated. The discovery and opening of a new frontier in a geographical sense is a typical irreversible phenomenon. The discovery was contingent, but having made the discovery, a people can do little but explore the discovered domain because they simply cannot undiscover it. Computing in education has a second frontier because an irreversible phenomenon of historic significance has been initiated that will deeply affect the potentialities and constraints of education. We can explore what lies beyond this frontier; we cannot return to a world in which the frontier does not exist.

Thus, our premise here is that we have initiated an irreversible action in cultural history in commencing to use computers for diverse activities in our culture. I think we can describe relatively precisely the nature of that irreversible action. To state it directly, the irreversible cultural action that we have initiated has two related components. The first consists in substituting a new form of coding—binary code—as the basis for storing and retrieving all the contents of our culture. The second consists in adding to the ancient cultural discovery of how to externalize memory outside the human mind, a very modern, portentous ability to externalize intelligence also outside the human mind. These two components are consequences of the introduction of digital technology, as noted above. What can be described in binary code becomes subject to recreation in multiple instances. People have initiated the description of all prior cultural achievements in binary code as well as the specification of diverse capacities for intelligent action in binary code. This second frontier is not simply technical, but deeply cultural. We have crossed it irre-

versibly. Discovering the possibilities beyond it will be a long, exciting journey.

Culture, as it has accumulated in history, is a vast store of externalized memory, memories that are put into things outside of human brains, into things that endure, inert but stable. Books and buildings, pictures and songs—all are memory externalized. Education has consisted largely in learning how to nurture and use a workable selection from this vast store. Up to now, to record things in externalized memory, a wide variety of coding mechanisms have been used—each medium of communication really has represented a different system for coding information outside of the inner memory of the human mind. The codes of writing are different from those of pictorial representation, which differ in turn from those of sculpture, architecture, still photography, or film. Culture has thus been divided into many domains of storage and retrieval, by the multiplicity of coding systems requisite for preserving it in external memory. With computers we are learning how to store all components of our culture in a more unified, single coding system, and the conversion to that new coding system has been irreversibly initiated.

To use computers in any domain, the material to be worked with must be described in binary code, for computers process information bit by bit. An irreversible action that we have commenced is to convert the coding of diverse cultural resources to binary code. First numbers and mathematical operations were expressed in binary code; then text; then images, complex chemical models, materials and structures, virtually everything. The conversion is happening and it is no more voluntary now than the completion of the sip of coffee that I initiated an instant ago. The time scale for its unfolding is far more elongated, and it may consequently, if understood, be susceptible to significant adjustment; but options for such corrective action are severely constrained, as with the burning house, and a wishful return to the *status quo ante* is not among them.

What is significant in this conversion of numerous different storage and retrieval mechanisms to one shared code is the eventual potential for cultural unification implicit in it. Currently, information is stored as text, or as mathematical expression, or as pictorial representation, or as recorded sound or image, or as physical structure or relief. The new coding that is going on translates each of these discrete systems to a binary base and its long-term cultural consequences are immense: From that shared base, root conceptions can be expressed in whichever representational form best suits the needs of the user. The mathematical relationship may start as a verbal proposition, be changed to a mathematical formula, and then to a dynamic, unfolding graphic curve.

Related to this change in the coding on which material memory can be based is another one that may be even more significant. We have lived up to now in an era in which culture, understood as the externalization of men-

tal achievements in objective things, has consisted exclusively in the remembrance of things. To be sure, we have long been able to use objectified memory to help instruct and discipline the intelligence. Thus the young study the recorded fruits of great intelligences with the aim of informing and activating similar capacities. The objectified memories, qua objects, were inert and dumb, primarily various marks on a page in one or another form of symbolic notation.

As we translate the stuff of culture into binary code and create more and more powerful tools for working with such binary code, we are increasingly able not only to store information in external objects, but to endow certain objects with the power to process information intelligently. The basic rule of digital technology is that insofar as we can specify something in binary code, we can recreate it. Increasingly, we are learning to specify intelligent operations in binary code, and insofar as we do, we can recreate in digital technology the forms of intelligence so specified. With this, we have irreversibly initiated the transformation of a culture of remembrance into a culture of intelligence. Until the historical present, people have learned how to store all the information they need in objects external to living minds; in the historical present people have begun to learn how to process intelligently the information they need through objects external to living minds.

All culture can be coded so that it can be operated on with digital computers, and the operation of digital computers is such that it will not only allow for the storage and retrieval of information through objects external to our minds, but will also permit the intelligent processing of that information in those external objects. Thus we have a more powerful tool for storage and retrieval than those hitherto available, one that further will provide its users with intelligently preprocessed information. We have crossed the frontier and initiated irreversibly a sequence of developments that will take a long time to complete, in which the cultural potentialities of these technologies are tried and tested. Some can regret the change, but they cannot reverse it; and others, like the authors who follow here, can welcome it and work to fulfill it.

The articles in this book consist of reports from beyond this pedagogical frontier, early explorations and attempted mappings of the way. In practical terms, the potentiality of binary code as a base code for multiple forms of representation will be developed steadily over time by those exploring graphical and multimedia computer-based educational resources. Likewise, the pedagogical problem encountered in shifting from a culture of memory to one of intelligence will consist of developing educational strategies through which people will learn how to control and direct the intelligent tools that will increasingly be available to them. These are the themes developed in the pages that follow. The tone is tentative and exploratory, a fitting tone for an effort the full possibilities of which will unfold only over a period of decades and generations.

Image Learning: Higher Education and Interactive Video Disc

BEN DAVIS
Massachusetts Institute of Technology

Traditionally, illustrations have been subsidiary to text in the higher learning. Visual information is being set free from its dependence on text and the consequences for education will be immense, if we can learn to exploit its possibilities.

Every age has its own peculiar set of interesting issues and ideas. Today, in our computer-enhanced, video-saturated society, this issue of visual computing in higher education has surfaced as a very significant one. That is why I have called this article "Image Learning: Higher Education and Interactive Video Disc." To get a sense of the centrality and import of this topic for the university in our day and age, a brief imaginary excursion through time will set the scene.

PREHISTORY

Let us go back in time to the first human beings, to prehistory. Imagine that the most intensely interesting topic for the Pleistocene age is "Higher Education and the Campfire." The new light-emitting technology of the campfire is having a profound effect on learning. It is providing heat, and meat never tasted better. It gives off light far into the darkness of night, keeping away predators. Everyone gathers around, discussing the fire, improving it a bit, staring into the flames, exchanging stories. The heat and safety of the fire allow a little wonder, allow a relaxed look at the stars. Bits of glowing bark blow up from the fire. Could the stars be distant fires? Beginning astronomy is now in session.

THE GREEKS

Time passes. It is 516 B.C. Mnemosyne is the Greek goddess of memory. Simonides is inventing the art of memory. He is equating the methods of poetry and painting. He is teaching that painting, poetry, and memory are intense visualization. In order to demonstrate this, spaces are designed with

visual details that elicit lines of poetry to the initiated. Carefully placed windows and small openings direct light onto these details. The topic of the day is "Higher Education and Memory Theater." Could it be that the mind has an eye? Do we remember the face and forget the name? Can objects be the repository of memory? Noumenon, phenomenon?

THE RENAISSANCE

We are in sixteenth-century Italy. The talk is of higher education and the printing press. Knowledge has been stockpiled in various libraries and monasteries where scholars dispense information to the elect. The illuminated texts are carefully handmade to create the illusion that light is coming from the page. The printing press has just been invented. Knowledge can be bought and traded like any other commodity. If you cannot read, just look at the pictures—every picture tells a story, does it not? Now the great teachings can go home with you, they are mass produced. The teacher-student relationship is changed. No single teacher can have read all the books, so education is distributed, it is "universitized," codified, filtered through institutions.

TELEVISION

What have we learned? We know we are a long way from higher education and the campfire. We know longer intuit directly from nature. The memory theaters of Simonides are pale marble monuments. Picture books are usually for art collections, travel, children, and fads. The fire has been replaced by television. *Tele-vision*—to see at a distance. Whatever is far away is close up. Whatever is too close can be made to seem far away. Pictures are literally in the air. We are entertained. Endlessly fascinated by this notion of images produced by light. If we choose, we can look at 1917 in motion or the tenth century in still images. We can see the aberration of black-and-white history, that brief period from the invention of photography in 1840 to the mid-twentieth century when color returned to images. And we can see advertising. We can see all about ourselves.

COMPUTERS

We can remember everything as well. We have computers. Powerful devices capable of astounding feats of memorization and organization. Electronic brains powerful enough to guide men to the moon and take pictures of distant planets. Enormous data banks that can be linked together to create electronic libraries more vast and encompassing than any single university. The computer began as a number-processing machine. It has now become a text- or word-processing machine as well.

THE VISUAL WORK STATION

What would happen if we combined optical, light-based visual memory and the computer?

John Berger, in *And Our Faces, My Heart, Brief as Photos*, reminds us that

> the visible has been and still is the principal source of information about the world. Through the visible one orientates oneself. Even perceptions coming from other senses are often translated into visual terms. (Vertigo is a pathological example: Originating in the ear, it is experienced as a visual, spatial confusion.) It is thanks to the visible that one recognizes space as a precondition for physical existence. The visible brings the world to us. At the same time it reminds us ceaselessly that it is a world in which we are at risk to be lost. The visible with its space also takes the world away from us. Nothing is more two-faced.[1]

What sort of a visual metaphor is this two-faced condition that lets light in but also can block it out?

WINDOWS

The window. Or better yet a series of windows, each with its own kind of vision, that can be studied individually or correlated by peering through simultaneously. The first window is for dialogue. It allows the explorer to engage in a search. We can call it a menu because it provides choices of action. The next window is for picturing; it is the representational visual memory. Then a text window for reference and entering information into the system from the keyboard. There is a graphics window, used for annotation, animation. These windows may be shifted, deleted, reordered, sized, moved, and iconified. At Project Athena this is called the X Window system. Developed at MIT in conjunction with Digital Equipment Corporation (DEC), the X Window is a network-transparent portable window system that allows applications to work seamlessly across various machine architectures and networks. It is a public-domain program originally written for Berkeley 4.2 UNIX. More than seventy computer companies and software developers have adopted X as a standard window manager, among them IBM, Sony, Hewlett-Packard, and Data General.

VISUAL MEMORY

Let us concentrate our attention on the picture window. How can picture data be entered into this window? It must come from a source that the computer can easily accommodate. It must be electric. In the late nineteenth century amazing changes took place, changes we take for granted as we enter the twenty-first century. Skyscrapers, elevators, the fountain pen, zippers,

the typewriter, airplanes, the electric light, all appeared as if by magic. The conversion of sound to electricity to sound again created the telephone, the radio, and the phonograph. Thomas Edison foresaw the phonograph as an instrument that would change mass education by bringing the voices of the great teachers to the ordinary person. In 1894 he invented the Kinetoscope, the motion picture projector, which he thought would replace textbooks. It was too expensive for education (as technology still is today) and instead became the premier instrument of entertainment. Edison actually wanted to combine the phonograph and the movie very early in his research "in the hope of developing something that would do for the eye what the phonograph did for the ear."[2] Motion picture records were never pursued, however. Instead, in 1926, sound was added to film.

Almost simultaneously with the developments in film, research on electronic image making was being undertaken by inventors like Philo Farnsworth and Vladimir Zworykin. They were inventing television. The main concentration was on cameras, cathode-ray tubes, and methods of transmission that have given us our modern television broadcast systems. John Loggie Baird, a Scotsman, was working in a divergent mode. He was attempting to make television pictures by mechanical means. He worked on the idea of inscribing television signals on a waxed phonograph record, and he called it Phonovision. This was in 1926. In 1935 he tried unsuccessfully to market Phonovision. Developments in broadcast television were simply too powerful and his vision of prerecorded television for the home or cinema was premature. It was not until 1963, after the far-reaching revolution in television, magnetic tape recording, and integrated circuitry, that the notion of video disc reemerged as a medium of interest. The random-access nature of the nonlinear disc was a compatible design for the computer.

LASER MEMORY

In 1916 Albert Einstein decided that he would spend the rest of his life attempting to understand the nature of light. His research led to his theories of quantum radiation. This work was fundamental to the invention of the laser (light amplification by stimulated emission of radiation). The laser is a device that excites electrons into emitting photons (the basic unit of light energy) of the same wavelength. It then pumps those photons into an intense amplified beam. The power of a million campfires. Focused on a tiny spot on a light-sensitive material, the laser can record and read microscopic holes, impressions that can be decoded into analog pictures. The storage density of this new medium, the laser video disc, is quite amazing. On one side of a 12-inch disc, 54,000 individual pictures can be stored along with stereo audio and digital data. The laser can access a single frame in 1.5 seconds or less. Compact Disc Read Only Memory (CD-ROM) is another kind of disc that is only 4.7 inches in diameter but can store 600 megabytes of digital

information or the equivalent of 200,000 pages of typed text. Entire encyclopedias have been placed on a single disc. Compact Disc Interactive (CD-I) is yet another innovation that proposes to eliminate the need for a computer by placing operating and applications information right on the disc as well as still photos, music, speech, graphics, and computer programs.

DIGITAL VIDEO

To bring our historical journey to the present, on March 3 of this past year, the David Sarnoff Research Center announced the creation of a full motion digital compact disc—in other words, a video disc with the same capacity as the larger analog disc but only 4.7 inches in diameter. This announcement is startling for another reason: The environment of the computer is digital. This new development fully realizes the notion of video becoming computer-compatible. It means that an inexpensive digital video player can be connected directly to a computer with very little trouble. It means in effect that all video will be computer-compatible, and that anything that can be placed before a video camera can be accessed by computer. It means that the visual world has a home in the computer. And the computer is inherently a learning machine.

LEARNING MACHINES

I say the computer is a learning machine rather than a teaching machine. Learning is a relational situation. We connect pieces of information to form a thought. Each of us has a special way of connecting information. The computer has the unique ability to imitate this mental process. Any student who moves from listening to classroom lectures to doing personal research has crossed into learning rather than being taught. The computer is a memory theater. It stores and displays information and is capable of being organized in an infinite number of ways. It remembers whatever is placed in it and can associate data in an enormous number of patterns. Anyone familiar with a word processor (as opposed to a typewriter) knows how fragments of text can be endlessly modified and rearranged and reassociated. Nothing is fixed until the computer activates the printing press.

PROJECT ATHENA

In an effort to better understand the effect of computers on learning, MIT, in 1983, created Project Athena. This five-year program was established to explore innovative uses of computing in the MIT curriculum. MIT faculty were concerned that for the most part only graduate-level students had access to computing power. In order to integrate computers into the undergraduate

program a large-scale effort would be needed, both technically and financially. Digital Equipment Corporation and International Business Machines agreed to provide MIT with approximately $50 million worth of hardware, software, technical support staff, maintenence, and networking. In addition, MIT raised $20 million for curriculum development. Twice a year faculty are invited to submit proposals to review committees. At the end of 1986, 111 projects were funded. It has become a large-scale laboratory to see how computing will fit into universities in the future. Its goal is a networkable coherence of machines and software. For instance, UNIX has been chosen as the operating system and X Window, developed at Athena, has become the chosen interface. Approximately 1,500 stations have been deployed in public work areas, living groups, laboratories, libraries, departmental areas, and special curriculum sites. Among these work stations are a cluster of very specialized machines, Visual Courseware machines. The visual work stations are demonstrating the combination of X windows, full motion color video disc, cable television, digital audio, high-resolution graphics, and CD-ROM, laser printers, and the very powerful 32-bit computers themselves (DEC MICROVAxII, IBM-RT).

There are ten courses currently under development: French, Spanish, German, and Japanese language; cellular biology (electron scanning microscope simulator), mechanical engineering (expert bearing selection), neuroanatomy of the human brain; and three architecture projects. Produced through the newly established Visual Courseware Group of Project Athena, these pilot projects represent an environmental approach to learning. (It is interesting to note that the subjects offered by MIT when it opened in 1865 were physics, mathematics, civil engineering, chemistry, French, and freehand drawing.) The student is interfaced with a multimedia networked station that is very much like a gateway into a new world of learning.

IMAGE LEARNING

In a world where pictures are literally in the air, the use of electronic imagery as a visualization tool seems obvious. What is not so obvious is how to grasp the tool. The use of new optical-information-storage technologies like laser discs acted on by powerful personal-scale computing machines makes possible enormous leaps in information correlation. Connected into interactive networks, these learning stations create a new model for higher education. All educational experience has a visual component; from physics to literature the mind's eye is at work. How a student visualizes and associates research becomes key to realizing the potential of the new technologies. By accessing libraries of text, video, audio, and computer graphics the student begins a journey into the collective memory. Image learning virtually means assuming the role of the artist, the person of integral awareness who creatively

looks, literally, for meaning. Each of us has a special way of learning, which can be coded into the memory of the computer. Each interaction with data bases can be noted and refined so that no subject is beyond a student's abilities. An art history interest can be applied to physics. You can comprehend whatever you are curious about because it can be shown to you the way you understand things. Everything is interesting; the world is open again to fresh insight.

QUESTIONS

Who will control this process? Who will decide content and structure? Who will have access? Is visual simulation a healthy idea? What will the inevitable marriage of education and entertainment mean? What will it mean to create knowledge gates that are independent of the campus and the institution? What role will the traditional teacher have in this environment? What will holographic visual memories mean?

These will be the questions for the next twenty years. The educator of today has responsibilities on a planetary scale. The seemingly innocuous introduction of something like interactive video disc into education suddenly tips a cultural scale. Interactive video disc is just a late-twentieth-century electronic campfire. It will allow a long gaze into our collective visual memories. What we see may astound us. The window can close, though. This is a two-faced world. While the window is open teachers must be responsible for what is seen. There is one basic factor that separates the educator from anyone else using these new light-based technologies: love of subject. The message of this new electronic learning environment is not the importance of machines or the possibility of a science of learning. It is that this is the time of the teacher.

VISUAL LANGUAGE LEARNING

The Visual Courseware Group at Project Athena is made up of creative individuals interested in visual language and with a passion for details about learning technologies.[3] By bringing this kind of sensitivity into a technological setting we hope to create a visionary computer environment, literally and metaphorically, for the students. What will this mean for the teaching of mechanical engineering, neuroanatomy, biology? Certainly the creators of the architecture projects already have an understanding of this process. We are discovering very quickly, however, that the language projects are vital to creating this visionary paradigm.[4] Language is environmental. It is not only concerned with words but with gesture, sound, body language, and facial expression. It is the dynamic fluid of cultural understanding and to this end all languages are in harmony. This is now a very small, fragile world that cannot risk misunderstandings.

Notes

1 John Berger, *And Our Faces, My Heart, Brief as Photos* (New York: Pantheon Books, 1984), p. 50.

2 Dagobert D. Runes, *Diary and Sundry Observations of Thomas Alva Edison* (New York: Philosophical Library, 1948), p. 64.

3 Visual Courseware Group: Dorothy Shamonsky, Matt Hodges, Russell Sasnett, Mark Ackerman, Stephen Wertheim, Ben Davis, and Mark Levine.

4 Athena Language Learning Project, funded by Annenberg/CPB Project, Janet Murray, director.

Moving Computing and Education beyond Rhetoric

ROBERT P. TAYLOR, NANCY CUNNIFF
Teachers College, Columbia University

Computers enable us to offer students distinctively alternative paths to certain goals, for instance, graphic representation in the place of verbal statement. Where such alternatives can be implemented, it becomes possible to test their comparative effectiveness with some rigor.

This article assumes that the supply of rhetoric about how computing can help or hurt education is more than sufficient for all conceivable purposes and that it is high time to move beyond that rhetoric in an effort to determine exactly how computing affects learning. Where computing is proven useful, it should be more thoughtfully and immediately applied; where proven useless, its application should be discouraged. Carefully designed and narrowly focused research is what is now needed. We argue that computing provides an effective and powerful way of providing alternatives for learners. It describes specific research and suggests how that research fits into the larger picture of what is needed to move beyond rhetoric.

Computing can help learning; however, how it does so is not well understood. Critical perspectives such as those voiced in an earlier issue of this journal[1] have obscured rather than clarified the issues by making their judgments on the basis of limited understanding. A realistic assessment of the role of computing in education can come only with a deep understanding of how to match the potential educational power of the computer with the needs of learners. It is the job of serious educators to define the questions to be asked and to conduct the research necessary to develop that understanding.

Since we believe appropriate research can identify how and to what degree computing can assist learning, we have undertaken a program of research along those lines. We summarize part of our work here to illustrate how research can further the understanding of the interaction of computing and learning. Our work involves a narrow area of education and a limited research focus, but the broader issue is an identification of how and for whom computing can be uniquely helpful. Our specific focus investigates a comparison of graphic versus textual representation of concepts. We hypothesized

and verified that graphic representation of some concepts is superior to textual representation for at least some learners. Since the computer is the principal means for making graphic presentation in the content area involved (computer science), acceptance of the hypothesis implies affirmation of the belief that the computer is a unique learning tool.

The computer can support many alternative representational forms, such as temporal compression, immediate access to information, simulation of movement and interaction, multidimensional graphic representation, and sound. In this article we concentrate on just one of those alternatives: the graphic representation of information as an alternative to traditional textual representation. We discuss the need for alternatives, and specifically address the potential of the computer for graphic representation in one specific educational context. We suggest that the traditional mold of education via text will be displaced only if there is empirical evidence that alternative representations are viable and even superior in at least some contexts, and conclude by giving a specific example of the superiority of graphic representation from the research we are conducting in the content area of teaching and learning programming. This computer science context is not chosen because it is the only one in which computing can be fruitfully applied or its potential contribution be clearly understood but rather because it is the specific area in which we work. Equally powerful cases could be made in other traditional subject areas, such as history, music, or art.

ISSUE: LEARNING, COMPUTING, AND CONCEPT REPRESENTATION

> I soon felt that the forms of ordinary language were far too diffuse . . . I was not long in deciding that the most favorable path to pursue was to have recourse to the language of signs. It then became necessary to contrive a notation which ought, if possible, to be at once simple and expressive, easily understood at the commencement, and capable of being readily retained in the memory.[2]

The need for alternative ways to represent ideas, concepts, and knowledge has long been acknowledged. Because different learners have differing capabilities for processing, understanding, and remembering material, no single mode of presenting concepts works equally well for everybody. Alternative representational modes are essential for ensuring that information will be understood and remembered by a wide range of learners. Formal education, however, seems to function on a radically different presupposition: Textual representation is adequate for nearly everything. There are many forces at work perpetuating the textual tone of formal education, such as:

1. Teachers teach as they were taught, thus the tradition self-perpetuates.

2. Few teachers have fluency in preparing graphic or auditory materials since teacher-preparation courses focus primarily on the development of skill in preparing textual materials and evaluation tools.
3. The preparation of nontextual materials is time-consuming and difficult.
4. School curricula involve the use of almost exclusively text-based materials.
5. The established methods for student evaluation are textual methods; decisions about a student's success, promotion, and acceptance to higher levels of education are based almost exclusively on the results of textual evaluations.

This article presents an argument that the overwhelmingly textual nature of school activity nourishes a serious imbalance in American education, one that probably prevents a considerable number of students from reaching their potential. The traditional practice of using text to the exclusion of other forms of representation for both the material to be learned and the means of assessing whether the student has learned that material is limiting and unnecessary. We argue further that computers can support alternative representations of learning materials, and that their use for such alternatives is essential to realizing the revolution in education predicted as a result of the arrival of computing.

Computers are one of the most powerful tools available to educators for the design and delivery of instruction based on a nontextual representation of material. However, although computers increasingly provide the means for infusing alternatives into our educational environments, most of their potential for so doing is yet to be tapped. This is so for several important reasons, including:

Well-established tradition dictates that schooling focus on the creation, manipulation, and processing of text-based materials.
Much educational software is merely a computerization of traditional textual materials.
Educators have limited experience with ways in which the computer can be used for nontraditional, nontextual applications.
There is little clear proof that alternative representations of learning materials improve learning.
The relatively high financial and human costs for the production and use of alternatively represented materials is not justified without such proof.

THE NEED FOR ALTERNATIVES: BREAKING TEXTUAL TYRANNY

The art of making pictorial statements in a precise and repeatable form is one that we have long taken for granted in the West. But it is usually

> forgotten that without prints and blueprints, without maps and geometry, the world of modern sciences and technologies would hardly exist.[3]

Although graphic representation was the primary mode of formal human communication for a long time before the advent of alphabetic symbols, since the invention of the printing press in the mid-fifteenth century, text has become increasingly central in formal communication. The ease with which print materials could be mass-produced and disseminated was alluring. Since early printing presses were capable of producing exclusively textual material, graphic representation was minimized (or at least let slip away) not because it was less effective than textual representation, but because reproducing text was more convenient, easier, and faster.

In consequence, we have become a culture whose dominant mode of communication in formal settings is print, a culture where education is almost synonymous with mastery of textual material. However, a vast amount of our information about people and things comes from nontextual, visual images; as television and video become more and more popular and widespread, this is becoming increasingly true. What we are seeing is a dichotomy between formal and informal communication, between how we typically learn in schools and how we learn elsewhere. We do not argue that graphic representation should again become the primary mode of communication. That, most certainly, would be unacceptable and foolish. We lobby for *balance.*

> We have all heard it said that one picture is worth a thousand words. Yet, if this statement is true, why does it have to be a saying? Because a picture is worth a thousand words only under special conditions—which commonly include a context of words in which the picture is set.[4]

There must be a middle ground where simple graphic images enhance and sometimes replace textual descriptions. Graphic representations can convey information rapidly[5] and can be remembered and recalled rapidly.[6] In a culture so deeply enmeshed in text as ours, graphic communication will not—and should not—*replace* oral and textual/print communication; rather it can and should *enhance* our communication by providing multiple representations of some information and alternative, simpler, or more direct representations of other information.

Provision of and teaching about alternative representations is too often ignored in the creation of teaching materials and curricular activities. Certainly, intuition suggests that graphics are often easier to understand than are textual descriptions of concepts or information. Western education, however, often proceeds as if the "written world" is sufficient for teaching about the real world.[7] A vast majority of student learning time is spent manipulating and creating text. New approaches all too often merely involve the use of different, but still text-based, materials. Even school textbooks that have many illustrations are often *used* as if they were exclusively textual because

many teachers are not fluent in the interpretation of graphically represented information, spending little time using such illustrations as charts and maps as focal points of instruction. At the same time, the problem self-perpetuates because students often do not know and are not taught how to interpret non-textual visual representations.[8]

Additionally, we have perfected the manipulation of text to such a degree that we "trust" textual representations of knowledge and rely on a learner's ability to interpret and produce text as the sole or sufficient measure of learning. In general, we use a learner's ability to understand, manipulate, and produce text as the measure of that learner's general ability and achievement. From the earliest years of school, evaluation of student understanding and achievement is textually based, and evaluation materials include measures of textual manipulation. Certainly, once we deem that a child *should* be able to read, achievement tests consist almost exclusively of text manipulation. This exclusive focus on text for evaluation suggests that the ability to manipulate text is the most important and most highly valued skill in American education.

An issue left unaddressed by the textual nature of most evaluative instruments is whether such evaluation really measures understanding. Does the ability to textually *describe* information or concepts universally indicate that a learner understands or can apply the information or concept? In many cases, educators assume that the ability to transform information into words implies mastery of the learning. There are cases where this may be so; however, there may be as many cases where this is a false test of a learner's understanding and ability to really use acquired knowledge. A more subtle point is: To what extent does our predilection for transforming all knowledge into text cause us to emasculate, alter, or degrade a concept into a variant that can be textually represented so that we can present it in our usual print form?

At any rate, successful students are, too often, those who have learned to create and manipulate text easily, rapidly, and purposefully. Those who have trouble with text are quite often unsuccessful in our educational institutions. The system offers little support to those who are not textually apt, ignoring or abandoning even those with highly developed alternative styles and aptitudes. Computing can and should be used to correct this imbalance.

THE ROLE OF THE COMPUTER IN GRAPHIC REPRESENTATION

Among the reasons why we traditionally make such minimal use of static graphics in education are the following:

1. Because of a predominantly textual training, teachers find it difficult to imagine how graphic representation might be useful in a particular presentation.
2. Because it is so time-consuming to prepare sufficiently accurate graphic representations, teachers sharply limit their use as part of prepared materials, in-class demonstrations, and lectures.
3. Because of the time it takes to generate most serious graphic work, students cannot be expected to produce much of it as part of regular homework assignments.
4. Because of the difficulty of accurately rendering any but the most trivial images and the consequent inherent danger that an image will be either misleading or confusing, even the occasional use of graphic representation is avoided as much as possible, by both teachers and students.
5. Because of the relatively high expense of including graphics in books, publishers constantly press authors to minimize their use wherever possible.

These are also the reasons we make no use of animated images.

The implications are obvious. Though an art student learns best by being able to recolor ten versions of the same colored design or by a retrospective analysis of all the versions a particular design traversed on the way to completion, the time required to produce these would be enormous and the cost of photographing or color copying every member of an extensive set of versions would be inconceivable within a school budget. Although the calculus student might understand the meaning of a function and its first and second derivatives best by seeing, for each of a group of related functions, the three curves corresponding to f, f', and f'' at twenty different value points, the time required for student or teacher to produce the appropriate rendering means it cannot be done, no matter how valuable it might be.

The computer's graphic capability changes this dramatically. Many computers now available, and more of those beginning to appear on the market, have the capabilities to directly and radically weaken reasons 2, 3, and 4, and to indirectly weaken 1 and 5. With software now available and increasingly with that beginning to appear, teachers and students can render all sorts of relevant images at high enough speed to make the analysis of a class of cases or flurry of versions perfectly reasonable as a basis for either class demonstrations or homework assignments. The accuracy of computer-generated images is well beyond what even the best teacher can do by hand and the capability to store a developmental sequence of versions or a set of cases is far in excess of the best set of notebooks any teacher or student has traditionally been able to maintain. There is no precedent for what is now available. It is a resource exclusively spawned by and supported through the computer. It awaits only more thoughtful application.

A REPRESENTATIVE RESEARCH CONTEXT: THE LEARNING OF COMPUTER PROGRAMMING

Intuitions about the communicative richness of graphics abound. It seems perfectly reasonable to believe that a picture is worth a thousand words, but without solid empirical evidence supporting that intuition, the argument for graphics as an alternative representational mode runs the risk of being a good idea that will never have an effect on educational practice. Although the research question of whether graphic representation is a viable alternative to textual representation could be studied in many contexts, we chose to investigate the questions in the context of teaching and learning computer programming because we have been teaching programming to beginners for several years, requiring them to learn and use two programming languages, one of which is graphically represented, the other textually.

Teaching and learning programming provides a good environment in which to investigate the effect of alternative representation for several related reasons. First, although experts have always sought out and relied on graphic representations of various sorts to clarify their arguments, programming languages have remained largely textual even though they incorporate ideas that are clearly representable graphically. There have been ongoing attempts to represent different aspects of programming graphically, but most are intermediate representational tools, ultimately requiring the programmer to write the program in a traditional, textual language. Thus, students have been forced to study programming in an environment based solely on symbolic, arbitrary, alphabetic notations. This type of textual environment for learning may be fine for experts and even for students who are inclined to see things textually, but for students without highly developed linguistic intelligence,[9] this exclusively textual approach is probably detrimental.

Second, a large number of students study programming on many levels of schooling. Additionally, there are many students who might wish to study programming but who are kept from doing so by the overwhelmingly symbolic nature of the subject matter and the monolithically textual form of available programming languages. A graphic representation provides the concreteness needed by some learners, helping them to grasp the abstraction of programming more readily.

Third, the growing interest in *visual programming* suggests that some developers, at least, believe graphic tools make it easier to understand the complex action of the computer.[10] If these tools help experts to understand complex systems, it seems reasonable to assume that a visual representation of programming constructs and logic would help novices understand programming more easily and quickly. However, while intuition and anecdotally recorded observation may have convinced many that graphic systems are viable, empirical research verifying that viability is badly needed.

Finally, and maybe most important of all, because computer science is a relatively "new subject" we are still trying to figure out *how* to teach programming: what materials and methods to use, and what sequence of conceptual presentation to follow. Because we are at such an early stage in the development of educational processes and products for the teaching of programming, educational researchers can and should address these questions, and then propose sound, empirically verifiable approaches for the development of alternative approaches.

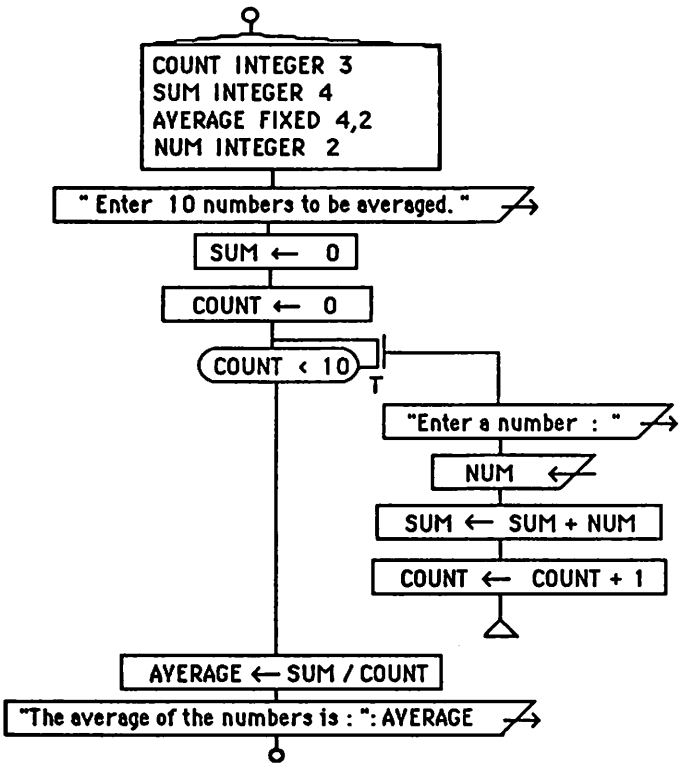
FPL: A GRAPHIC PROGRAMMING LANGUAGE

For several years we have been teaching programming to novices using First Programming Language (FPL), a graphically represented programming language under development at Teachers College, Columbia University. The testimonies that FPL is learnable, by a large number of novice programmers with little mathematics or science background, reinforced our original intuition that a graphic representation is useful, and have also suggested that we should seek their verification empirically. FPL is fully described elsewhere,[11] so the following comments are intended only to convey the barest sense of how it compares to a textual programming language.

FPL is a graphic representation of classical programming designed for teaching programming to novices. It uses icons and spatial arrangement to graphically represent the programming actions. There are eleven FPL icons; each represents a specific programming action. Eight include programmer-inserted text, the variables and constants of the program; three include no text. The icons supersede the "reserved words" or instructions of text-based programming languages and thereby embody flow of control and logic.

FPL currently runs on computers in the IBM-PC family. It is not merely a computer-based flowchart to be used only for planning a program that must itself be rendered in a traditional, textual language to actually run. FPL is a fully functional language that provides programmers with a graphics-based, spatial environment for the creation of executable computer programs.

The major difference between FPL and other programming languages is its graphic representation. Programs in classical programming languages must be read in a linear, proselike fashion even when the action of the program is nonlinear. Because of its unique spatial layout of connected symbols, FPL allows a reader to see her or his program as a map, in a format that more directly emphasizes its logical structure. Figure 1 presents a typical beginning program in FPL and, for comparison, its counterpart in Pascal.




```
PROGRAM test ;
 VAR
   count : INTEGER;
   sum : INTEGER;
   average : REAL;
   num : INTEGER;
BEGIN
   WRITELN('Enter 10 numbers to be averaged.');
   sum := 0;
   count := 0;
   WHILE count < 10 DO
      BEGIN
         WRITELN('Enter a number : ')
         READLN(num);
         sum := sum + num;
         count := count + 1;
      END;
   average := sum / count;
   WRITELN('The average is : ',average:4:2);
END.
```

Figure 1. FPL Program with Pascal Translation.

INVESTIGATIONS OF A GRAPHIC ALTERNATIVE

Within the context of teaching programming and investigating the relative merits of a graphic representation, the exact subcontext we chose was: to determine whether for some novice learners a graphic representation of the conceptual material of computer programming is more effective than a strictly textual representation. Our work investigates two "informationally equivalent"[12] representations of programming in an attempt to provide sound empirical verification that in one specific educational context graphic representation of concepts can be superior to traditional textual representation. Just as there are many contexts in which the viability of a graphic representation could be studied, so there are many facts within those contexts that can be investigated. Our current work focuses on two facets of programming, finding bugs (errors) in programs and comprehension of programs written by others.

Bugs in Programs Written by Novices

Our early studies of the effect of FPL's graphic representation focused on the presence of bugs (errors that result in unwanted or unexpected behavior by the computer during execution) in programs written by novices. In two different studies[13] we catalogued bugs in FPL programs written by our students and compared those bugs with observations of novices' Pascal programs made by Soloway and his colleagues at Yale University.[14] We analyzed the types and frequency of *logical* (nonsyntactic) bugs in these programs. Logical bugs reflect errors in or misconceptions about problem solving rather than errors in the syntax of the programming language. Since the FPL and Pascal programs studied were solutions to the same programming problems, we could compare problem solving across two representational modes, graphic and textual.

We found that, although there seem to be some types of bugs that are language-independent and are closely allied to instruction, others appear to be clearly affected by the programming language used for solving the problem. For example, students writing programs in FPL did not misplace program statements, or *icons* (as they are called in FPL), while students writing in Pascal frequently did so. In the same vein, certain types of program statements were often missing in the Pascal programs while this was not the case in the FPL programs.

We speculated that the spatial and graphic representation of programming constructs allowed novices to *see* the structure and scope of the program more clearly, resulting in fewer errors. It may be that the individual elements of the graphic representation were more memorable, thus resulting in more correct construction of programs. It is also possible that in learning elementary programming plans,[15] the spatial layout of FPL makes the placement of plan elements such as variable updates and running totals easier to remember.

Program Comprehension

In another series of studies we investigated whether novice programmers learning both FPL and Pascal comprehend programs in one language more accurately and/or more rapidly than in the other.[16] Using an on-line reaction time system, our subjects viewed a series of short program segments and answered comprehension questions about those segments. Each program segment was coded in both FPL and Pascal; the logical structure of the two was identical but the context and details differed to mask the repetition. The same questions were asked about both versions of each segment. To provide a more discriminative context for data interpretation, the subjects' visual and verbal aptitudes were measured using standard instruments.

We found that comprehension of FPL program segments was significantly more rapid than was comprehension of Pascal versions of the program segments. The results were even more pervasive than we had anticipated; while we hypothesized that FPL would be comprehended more rapidly by subjects with high visual aptitude, we found that for almost all subjects, regardless of visual aptitude as we measured it, this was the case. Thus, the findings seem to indicate that comprehension of short program segments is more rapid when the segments are graphically represented.

Comprehension speed is important only insofar as the programmer is able to comprehend *accurately*. Therefore, accuracy was also measured in these studies. So that comprehension reaction time could be measured with confidence, the questions were deliberately designed to be answerable by the subjects involved in the study. Thus, we were not surprised that 91 percent of all questions were answered correctly. This relatively high level of accuracy with respect to both languages involved confirms subject facility with both, minimizing the possibility that results were due largely to subjects' being more knowledgeable in the graphic language.

Although the data on incorrect answers were small, they were interesting. Of those questions that were answered correctly for one version of the segment and incorrectly for the other, 79.7 percent were incorrect responses to the questions about the Pascal segment while only 21.3 percent involved incorrect answers to FPL segments. Thus, the results strongly support our hypothesis that graphically represented programs would be comprehended both more rapidly and more accurately than their textually represented counterparts.

For novices, comprehension is a critical aspect of learning programming, integrally involved in program construction and debugging. Helping novices with the incremental steps involved in the larger task we refer to as programming will certainly help allay frustration and avoid failure. More rapid and accurate comprehension will certainly contribute to learning, affecting both program writing and, most certainly, program debugging. Our findings suggest that the graphic representation helps novices assess certain aspects of a

program segment more rapidly and accurately, thus leading us to conclude that a graphic representation is an important alternative, at least for novices in this domain.

CONCLUDING COMMENTS

This article argues that educators can and should use the computer to provide alternative ways of representing knowledge. In particular, our research focused on using the computer to implement graphics as an alternative to the traditional textual representation of concepts. The results suggest that a full understanding of the effect of computing on education can only come from rigorous and extensive research exploring the relationships between learning and the alternative forms of representation the computer offers.

We began with the assumption that without empirical evidence proving its effectiveness, the computer will serve only as a platform for educational rhetoric, and offered evidence that the computer can positively affect learning, illustrating the argument by reference to FPL, a graphically represented programming language used to teach programming to novices. We summarized some narrowly focused research conducted in one subsection of a single academic subject, computer science. Though such a minute investigation does not provide all of the evidence needed, it certainly does begin to prove that when used to provide alternative representations of material, the use of computing can affect learning. In particular, when used to present certain kinds of material graphically rather than textually, the computer can help a majority of learners comprehend some aspects of that material faster and more accurately.

Limited though it is, we are convinced this is the sort of careful research that must be done if we are ever to fully appreciate exactly how computing can and cannot help us learn. Clearly a very large agenda of research lies ahead. However, as Shneiderman[17] has pointed out with respect to another aspect of human-computer interaction, each small piece of research fits as "a tile into the mosaic" of a large agenda.

Notes

1 "The Computer in Education in Critical Perspective," *Teachers College Record* 85, no. 4 (Summer, 1984): 539-639.

2 From *On a Method of Expressing by Signs the Action of Machinery*, in *Charles Babbage and His Calculating Engines: Selected Writings by Charles Babbage and Others*, ed. P. Morrison and E. Morrison (New York: Dover Publishing, Inc., 1961).

3 M. McLuhan, *Understanding Media* (New York: McGraw-Hill, 1964), p. 145.

4 W. Ong, *Orality and Literacy* (New York: Methuen, 1982), p. 7.

5 K. N. Lodding, "Iconic Interfacing," *IEEE Computer Graphics and Applications* 18 (August 1983): 11-20.

6 A. Pavio, *Imagery and Verbal Processes* (New York: Holt, Rinehart & Winston, 1971); and L. Standing, "Learning 10,000 Pictures," *Quarterly Journal of Experimental Psychology* 25 (1973): 207.

7 W. H. Huggins and D. R. Entwisle, *Iconic Communication* (Baltimore: The Johns Hopkins University Press, 1974).

8 R. Arnheim, *Visual Thinking* (Berkeley: University of California Press, 1969).

9 H. Gardner, *Frames of Mind* (New York: Basic Books, 1983).

10 See the August 1985 special issue of *IEEE Computer* for an in-depth discussion of some current projects involving visual programming. Of particular interest are R. B. Grafton and T. Ichikawa, "Visual Programming: Guest Editors Introduction," p. 6-9 and G. Raeder, "A Survey of Current Graphical Programming Techniques," p. 11-26.

11 See R. P. Taylor, *Programming Primer* (Reading, Mass.: Addison-Wesley, 1982); idem, *FPL: Graphical Representation of Classical Programming* (New York: Teachers College, Columbia University: Department of Communication, Computing and Technology in Education, 1986); and idem, N. Cunniff and M. Uchiyama, "Learning, Research and the Graphical Representation of Programming," in *Proceedings of the Fall Joint Computing Conference* (New York: Association of Computing Machinery, 1986), pp. 56-63.

12 J. H. Larkin and H. A. Simon, "Why a Diagram Is (Sometimes) Worth Ten Thousand Words," *Cognitive Science* 11 (1987): 65-99.

13 N. Cunniff, R. P. Taylor, and J. B. Black, "Does Programming Language Affect the Types of Conceptual Bugs in Novices' Programs? A Comparison of FPL and Pascal," in *Human Factors in Computer Systems: Proceedings of CHI '86* (New York: Association of Computing Machinery, 1986), pp. 175-82; and N. Cunniff, R. P. Taylor, and S. Taylor, "The Effect of Programming Language on the Conceptual Bugs in Novices' Programs: A Comparison of FPL and Pascal," in *The Role of Language in Problem Solving II*, ed. B. W. Hamill, R. C. Jernigan, and J. C. Boudreaux (New York: North-Holland Publishers, 1987), pp. 391-410.

14 See W. L. Johnson et al., *Bug Catalogue I*, Tech. Rep. No. 286 (New Haven, Conn.: Yale University, Department of Computer Science, 1983); J. C. Spohrer et al., *Bugs in Novice Programs and Misconceptions in Novice Programmers*, Tech. Rep. (New Haven, Conn.: Yale University, Department of Computer Science, (1984); and J. C. Spohrer, E. Soloway, and E. Pope, "Where the Bugs Are," *Proceedings of CHI'85* (New York: Association of Computing Machinery, 1985), pp. 47-53.

15 E. Soloway. "Learning to Program = Learning to Construct Mechanisms and Explanations," *Communications of the ACM* 19 (September 1986): 850-58.

16 See N. Cunniff, *The Graphical Representation of Programming: The Effect on the Comprehension of Novice Programmers* (New York: Teachers College, Columbia University, in preparation); idem and R. P. Taylor, "Graphics and Learning: A Study of Learner Characteristics and Comprehension of Computer Programs" (Stuttgart: *Proceedings of INTERACT '87*, 1987), pp. 317-22; and idem, "Graphical *versus* Textual Representation: An Empirical Study of Novices' Program Comprehension," in *Empirical Studies of Programmers, II*, ed. G. Olson, E. Soloway, and S. Shepard (Norwood, N.J.: Ablex, 1987).

17 B. Shneiderman, "Empirical Studies of Programmers: The Territory, Paths and Destinations," in *Empirical Studies of Programmers*, ed. E. Soloway and S. Iyengar (Norwood, N.J.: Ablex Publishers, 1986), pp. 1-12.

Interactive Video Systems: Their Promise and Educational Potential

CARLA SEAL-WANNER
Teachers College, Columbia University

Interactive video is an exciting technology, but why might the addition of interactivity to moving images that are already ubiquitous for the young with movies and television make them more educative? Here are three hypotheses pertaining to this question.

It will not be long before children's access to computer-based interactive video systems (IVS) is as widespread as their use of the VCR. In schools, children may soon learn French from an interactive video system by completing a story in which they define their own role and can influence the outcome of the narrative. In a museum setting, the child may become engaged in a problem-solving exercise to test scientific hypotheses through laboratory simulations. In recreational centers, airports, or even the Paris metro, children may find video discs housed in information kiosks that allow them to "walk" around different parts of the world, get directions, or learn about the history of the city they are in. In department stores, the young shopper might view an entire line of clothing before selecting the designer T-shirt of his or her choice from a point-of-purchase video-disc console.

Whether children's introduction to interactive video is in a formal or an informal learning context, they will quickly discover that, among other engaging characteristics, this new technology has two features that are intrinsically satisfying to any user: An IVS lets the user be the boss and it responds almost immediately to the user's instructions. Children will rapidly learn that they are in charge—to get the system to react, they must act. Their actions cause the system to respond with a multiplicity of visual, auditory, and text-based messages. Some interactive video systems may partially direct the user's activity while others may be completely open-ended. At their best, interactive video systems will provide an environment in which children can learn to learn in an active, self-directed way.

Interactive video systems are a family of devices that have in common the coupling of motion video and audio with the interactivity potential of personal computers. This is typically implemented by a 12-inch laser video disc, capable of storing up to 54,000 still frames of video or up to 30 minutes of full motion video and audio. Each image can be individually addressed by

a suitably interfaced personal computer. The computer controls the sequencing of still-frame and motion video and audio in response to the user's inputs. From the user's point of view, a typical application presents itself as an ordinary television screen, plus one of a number of user-response devices, such as a keyboard, mouse, joystick, light pen, or touch-sensitive screen.

Given the increasing frequency with which such systems are likely to populate the child's world, educators need to begin asking how children's orientation to learning might be changed by using interactive visual courseware, participating in a simulation of a scientific experiment, being a surrogate traveler before actually setting foot in a part of the world that is new to them, or collecting "data" for decision making about how they will spend their allowance. Is it plausible that these experiences could change a child's expectations about the learning process and his or her role in it?

The current enthusiasm for this new technology has developed out of a long history of efforts to employ technology in education. Each generation of learning technology has tended to reflect a different pedagogical theory. Early learning machines derived from a behavioristic approach to education, which could be implemented on the simple drum devices then available. Although undeniably useful for the rote drill employed in vocabulary learning or memorizing the multiplication tables, this technology fell far short of stimulating those more active learning processes that educators since Dewey had believed necessary to attaining higher competencies. In effect, this tradition paved the way for the development of great expectations for increasingly interactive computer-assisted instruction (CAI) and interactive video systems, both of which promised in various ways to provide a more active learning environment. Until the relatively recent advent of this new technology, the practical problem of providing the kind of individualized instruction necessary to encourage the development of more complicated thinking skills proved insurmountable. There simply were too few teachers or hours in the day to distribute equitably among the students who needed individualized attention. The new interactive learning technologies offer a possible solution.

The solution many educators and psychologists hope this new technology will provide is an automated system of instruction that truly encourages inquiry-oriented learning. Inquiry-oriented learning theorists from Dewey to Bruner and the modern cognitivists view instruction as a way of encouraging students to become "active constructors of knowledge, of knowledge as open and evolving, of academic learning as exciting and vital, and of teaching as a stimulus to curiosity and a model in inquiry."[1] The goals of this kind of instruction seem to be highly correlated with the particular characteristics of computer-assisted interactive video, in which the sequence and selection of messages are determined by the user's response to the material. The purpose of this article is to describe how the various interactive features of these

electronic learning environments have the potential to encourage the kinds of learning that these educational reformers have long advocated.

I have some ideas about why computer-based interactive video might match some of these instructional objectives. I have framed these ideas as hypotheses because they can and should be put to empirical test in the research and development process.

HYPOTHESIS 1: PROGRESSIVELY MORE INTERACTIVE VIDEO INSTRUCTION, YIELDING INCREASINGLY GREATER LEARNING

The interactivity potential of an IVS is usually discussed in terms of the responsiveness of the system to the user. Responsiveness could be measured by two key attributes of control available to the user. First, how often can the user interact or intervene? The frequency of control in an IVS lies on a continuum, ranging from discrete opportunities to intervene, where the opportunities are limited (e.g., a true/false question), to continuous interaction, where the user is unlimited by the system and can exert control at virtually any time (e.g., in a flight simulation). The more advanced systems provide a richness of control that models the complexity and flexibility of interaction with the real world. For example, in simulations of laboratory experiments, motion decomposition for athletic training, or surrogate travel, the time intervals are finely graded, and the experience is rich with opportunities to intervene and explore the particular learning task. Therefore, these educational simulations should closely approximate the unlimited opportunities the individual would have for experimentation if he were really learning how to fly a plane, figure out how to get from one place to another, or ski a slalom course—without, of course, the dangerous consequences and high cost of real experimentation in these domains.

To determine whether a more flexible system promotes greater learning, the particular options for control available to the user should be evaluated. What methods for exerting control are available to the learner at each intervention point? The various formats of interactivity designed for these systems may serve to query learners, provide relevant feedback and practice on important concepts, and generally promote active engagement in the learning process. For example, the child who becomes a surrogate traveler in the ancient Maya ruin of Palenque via Bank Street College of Education's interactive video system can explore the archaeological site by changing speed of travel, point of view, and direction, by stopping to enter and wander through hallways and stairwells, by accessing maps or using other cartographic representations, by changing scale and zooming in on locations or objects of interest, and by relying on pictographic cues to search for additional information.[2] In general, an IVS offers users the option to change difficulty level;

access on-line assistance, glossaries, and data bases; and change the sequence of the learning modules on their own schedule.

All of these options allow the learner to be in control of the learning process by discovering how to navigate within the learning environment in order to initiate queries and influence the responses of the IVS and the information obtained from it. The more opportunity for interaction, the more the learning experience becomes a process of self-directed inquiry. Sam Gibbon, director of Bank Street College of Education's Multimedia Project in Science and Mathematics, argues that the responsiveness of the system will determine children's active engagement in their own education:

> The flexibility of response seems to me to be critical in determining how successful these learning environments will be. They must, I believe, be learning environments, rather than teaching environments. That imposes some constraints on the mode, the style, and the content of the responses that the environment makes to the learner's actions.[3]

In designing electronic learning environments the emphasis should be placed on developing a pattern of interactivity that reinforces whatever unique path of inquiry the child chooses.

Despite the obvious pedagogical value of this idea, very little empirical evidence exists to support the claim that the responsiveness or interactivity of an IVS increases learning. So far, only one study (that I am aware of) directly supports the notion that computer-based interactivity with video instruction facilitates certain types of learning.[4] In a design that compared students' recall and comprehension after video instruction with increasingly interactive exercises, learning improved with increased interactivity. Interactivity was incorporated in the form of embedded questions, feedback for correct and incorrect responses, and video presentations that branched to new material on repeated video segments. This study may not have taken full advantage of the capacities of an IVS but the results are a preliminary confirmation that interactivity seems to enhance some kinds of learning. Additional indications come from anecdotal reports of research and development efforts to create and test prototype interactive video systems for various educational purposes.[5] However, these systems are still in the formative stage and the evaluations remain inconclusive.

Eventually, we need a testing program that explores the full matrix of interactive formats crossed by different types of learning. There is no reason to believe that one interactive format will be superior for all types of learning. There may be different formats that are optimal for different types of information, different individual learning styles, and children of different ages. A full assessment of interactivity should attempt to make a map of these differences.

HYPOTHESIS 2: THE RICHER AND MORE COMPREHENSIVE THE INTERACTIVE ENVIRONMENT, THE GREATER THE RESULTING LEARNING

A principal characteristic of interactive video learning environments is that information is available through a variety of presentation styles not previously possible in any one system. Information can be presented through images as well as alphanumeric symbols, auditory as well as visual recordings, and dynamic as well as static imagery. The potential richness of this electronic learning environment depends on the unique combination of these characteristics and attributes. The option of interchangeably selecting types of images, or combinations of information presentations, for given learning tasks separates interactive video from both CAI and video-based instructional technologies.

Some may think that these features are only motivational devices—a fancy way to get the learner's attention. Although the sheer novelty of this combination of features may simply hold the user's attention and motivate interaction in the beginning, these features have a more important function as the user's "gee whiz" reactions wear off. In fact, we know from television research that although viewers initially attend selectively to the salient perceptual formal features of a program on television (e.g., special video effects, fast pace, pans, zooms, sound effects, and loud or lively music), familiarity with these features eventually gives rise to a viewing pattern in which the nonsalient features of the production (e.g., reliance on dialogue to carry a story line, inserts with informative material) become the focus of attention and determine comprehension.[6] Even younger viewers (over seven or eight years old) begin to search for the more substantive and informative aspects of a program. Although there is no research that documents this phenomenon for interactive video, I would predict that the interactivity component would promote a similar change in the way the user processes and utilizes information. With increased exposure to an IVS, discontinuous and impulsive attentional patterns that might be observed at first should give way to a kind of information-gathering activity that is instrumental rather than consumatory and is guided by internally generated goals rather than by external sensory events.

At a fundamental level, interactive video employs its unique combination of multimedia features to effectively focus attention on intended learning of important information. This technology has a myriad of engaging methods available for creating instructional sequences that allow users to dynamically test, revise, and broaden their thinking by interrelating relevant pieces of information. At its best, IVS can utilize these techniques to provide students with two essential ingredients of effective instruction: First, an IVS can place facts in meaningful contexts; second, it can provide students with the oppor-

tunity to create meaningful contexts on their own. Both of these functions deserve elaboration.

1. *Giving a fact a place to go.* Good teaching depends on the ability of the instructor to present information, facts, and concepts in such a way that students will remember them. To achieve this goal, teachers often supplement their lectures with examples that bring the ideas they present alive by making a concrete connection. Some teachers provide examples, or analogous situations, by reading or creating stories, showing a video or film segment, or using a demonstration on the computer. We have all had at least a few of these experiences of learning by example in which a teacher creates a meaningful context for a fact, or presents a particularly revealing analogy that causes the proverbial light bulb to go on. As the inventive software designer Tom Snyder put it in his recent book, "When a fact has a place to go, when it fits into a context that makes sense, however fanciful, it will curl up like a child in pajamas and stay a long time."[7]

Interactive video systems have a multidimensional capacity to house facts, concepts, and even the kitchen sink, in meaningful contexts. Information can be presented in narrative accounts, simulations, interviews with experts, graphic displays, and game sequences—to name but a few of the options available. To take one example of how these contexts can be employed, in the French-language video disc developed by MIT's Project Athena, the learner is called on to help a young man find an apartment to rent in Paris (a well nigh impossible task).[8] The learner must make decisions for the would-be renter and engage in various problem-solving exercises—all in French. To be of help, the student needs to remember the details of the young man's situation, as well as the vocabulary necessary to respond appropriately to his problem. Therefore, knowledge of vocabulary is embedded in a meaningful context in which it is used to achieve a realistic goal. Other examples of this technique abound in such disparate subject areas as mathematical problem solving, archaeology, history, biology, physics, and the formal analysis of film.

One of the most resilient facts to emerge from the psychological study of memory is that items incorporated in a meaningful context are remembered better than items presented in an unstructured list.[9] Clearly, interactive video courseware has enormous potential to make use of this psychological fact for the improvement of education.

2. *Allowing students to create their own contexts.* The unique features of an IVS allow for another important pedagogical goal to be realized—that of individualizing instruction. An IVS has the potential to provide the student with an opportunity to fit a lesson into his particular interests, instructional needs, and skills for assimilating knowledge. In effect, the variety of methods for presenting information allows students to discover their own personal paths of inquiry and preferred styles of knowledge representation.

Again, this is not a new idea in education. In *Emile,* Rousseau set forth the then radical notion that education should nurture a child's natural capabilities. He argued that the goal of education is to teach the child what is useful in a way that has some meaning for the child. For instance, Rousseau made this suggestion about how Emile should be taught geography:

> In any study whatsoever, the symbols are of no value without the idea of the things symbolized. Yet the education of the child is confined to those symbols, while no one ever succeeds in making him understand the thing signified. You think you are teaching him what the world is like; he is only learning the map. He is taught the names of towns, countries, rivers, which have no existence for him except on the paper before him. . . . After two years' work with the globe and cosmography, there is not a single ten-year-old child who could find his way . . . by a map about the paths of his father's estate without getting lost.
>
> Emile's geography will begin with the town he lives in and his father's country house, then the places between them, the rivers near them, and then the sun's aspect and how to find one's way by its aid. . . . Let him make his own map, a very simple map, at first containing only two places; others may be added from time to time, as he is able to estimate their distance and position. You see at once what a good start we have given him by making his eyes his compass.[10]

Either by means of a situation that is created by the IVS, or by the user himself, a wealth of opportunities should become available for the student to construct his or her own context for ordering the information that is gathered and representing the knowledge that is learned.

An additional advantage of this technology is that it provides a range of options for responding to the diverse learning styles of students. For example, in a biology application, one student may use the text and graphic capacities of the system to collect and display data from a simulated experiment, while another student might choose to present this evidence by assembling a montage of still and motion video images that demonstrates the phenomenon observed. In this way, students whose ability to demonstrate mastery in a particular subject area is limited by their lack of understanding of how to represent knowledge in an abstract way may find that they can express what they have learned in a more congenial presentational style. In mathematical problem solving, for example, a student might not be able to write the algorithm for a problem solution, but might be able to represent the result in a narrative account on video. On the other hand, some students may be more confident about representing solutions to problems in a conventional procedural method. This option would be equally available on an IVS. These multiple formats also allow students to go back and forth between the formats they are comfortable with and equivalent representations in formats they

find less congenial. In this way they can build up familiarity with formats they previously found uncongenial and eventually acquire mastery over them.

Another logical consequence of this process is that by providing students with different ways to represent the knowledge they have gained we can assume that they will eventually have to consider the representations of others. This suggests the possibility of encouraging another kind of interactivity. Ideally, this exercise can lead to collaboration with others as students attempt to construct the most convincing argument for representing what they have learned.

HYPOTHESIS 3: LEARNING HOW TO MASTER AN IVS SHOULD ENCOURAGE CHILDREN TO TAKE A MORE ACTIVE PART IN THEIR OWN EDUCATION

As I have argued in support of hypotheses one and two, there is good reason to believe that reciprocal exchange between either the student and the teacher or the student and an IVS promotes learning. Proponents of inquiry-oriented learning argue further that it is particularly important for students to learn to initiate these interactions on their own. In the ideal educational environment envisioned by these theorists, reciprocity is achieved when the teacher provides access to knowledge and resources as they are requested by the student. Unlike the traditionally conceived teacher-student relationship, where the teacher shapes the child's paths of inquiry, the responsibility for initiating the learning activity in inquiry-oriented learning is shifted from the teacher to the student.

A well-designed IVS has the potential to realize this kind of student-initiated learning experience. The electronic learning environment provided by an IVS resembles inquiry-oriented learning because the responsibility for initiating inquiry is also shifted to the student. However, with this learning technology, the responsibility for taking initiative is twofold. First, the student must learn to pursue knowledge embedded in the instructional system by developing a personal path of inquiry. This kind of learning might be thought of as *epistemological* mastery, in which the student takes responsibility for learning how to learn. In addition, the student must learn to manage the dialogue with the IVS in order to obtain information from the system. This kind of mastery is obviously more *technological.*

All interactive video systems require a certain degree of problem solving to figure out how to navigate through the system and manipulate and control the flow of information most effectively. Generally, students gain technological knowledge through trial-and-error experimentation that eventually leads to the development of increasingly efficient strategies for accessing the tools and resources of the system. By gaining this degree of control over an IVS,

students may experience a certain kind of empowerment because they become authorities on how the system can best be used to serve their own learning needs. It would seem that becoming expert in this domain would lead students to direct their learning more confidently and creatively. Although this electronic learning environment can never supplant human interaction with a teacher, removing the teacher may lead to a degree of independence that even the ideal tutor-tutee relationship cannot achieve.

All interactive learning technologies offer the possibility of gaining mastery in the sense of being able to repeat an instructional sequence until stable performance is achieved and demonstrated. However, I suspect that carefully designed interactive video systems have the potential to encourage users to develop epistemological and technological mastery that goes beyond simply ensuring a repeatable performance. These kinds of mastery are far more important to a child's education in the long run than just getting the correct answers on a test. Clearly, the responsibility is on the instructional designers to ensure that the system is designed to facilitate the attainment of these kinds of mastery. A system that is insensitive to user initiative could result in a passive reaction rather than an active and exploratory attitude toward mastering the system on these two levels. The potential for nurturing that attitude is clearly latent in the technology, but it must be appropriately used.

It is the characteristics of interactive video systems discussed above that underlie the basic assumptions about the instructional effectiveness of this technology. Whether these assumptions can be validated is not presently known. As I have indicated, despite the fact that many endorsements of the instructional efficacy of interactive video have been offered, very little empirical evidence exists to back up these claims. Clearly, an extensive formative and summative research program, involving the systematic development and testing of this new tool for learning, is called for. Such a research agenda must include an evaluation of the effectiveness of interactive video systems for presenting different approaches to learning and different subject matter to students with a variety of different learning styles. In particular, studies should focus on gender and developmental differences in the use and appeal of the various components of this technology.

In a sense, the compelling face validity of interactive video education has preempted the developmental research needed to evaluate its true effectiveness. In particular, while the learning environment of an IVS seems likely to encourage students to be active inquirers rather than passive recipients of knowledge, we still know little about how IVS environment can best achieve this goal. Further, cognitive psychologists would argue that to demonstrate the effectiveness of an IVS for enhancing this kind of learning, we must determine if this method of instruction can promote automaticity, the building

of associations, the generation of meaning, and the transfer of skills from one environment to another. While it is difficult to demonstrate the impact of even conventional instruction on the development of these higher-level thought processes, research on this new technology should seek to determine the ways in which the mode of instruction possible with an IVS might encourage the kind of learning that would lead to the development of these skills. We are a long way from having answers to these questions, but they should be central to any research-and-development efforts.

Testing prototype interactive video systems is important for addressing the pedagogical issues raised in this article, as well as other practical and design-based questions raised by this new technology.

First, as prototype systems are developed, their potential for contributing to equity in the classroom should be a focus of research as well. This research should attempt to determine how this technology can benefit children in classes where instruction may be targeted too high or too low for their individual abilities or knowledge, teaching may be inadequate, teachers have a limited amount of time to spend with individual students, and teaching may not be race-, class-, or sex-neutral. The impact of this technology on promoting equity across educational settings could be dramatic. This issue deserves careful evaluation.

Finally, I think research should focus on some design questions. Because interactive video systems are the precursors of the more complex and elaborate multimedia learning environments of the *near* future, we need to anticipate the questions that design teams will be asking. The interfacing of these systems will substantially increase the number and type of learning tools, archival material, and data bases available to students for pursuing their studies independently and creatively. These systems will also include the potential for connecting (through computer networks) with other multimedia environments. What information will aid the creative imagination of the instructional designers, computer specialists, and video and computer graphic artists who will assemble these multimedia environments? To create an informative and engaging interface designers will need to understand the affective needs and aesthetic reactions of different students to these electronic learning environments. Specifically, it is important to gain insight into how these sensibilities interact with the learning process. In anticipation of increasingly complicated interface designs, research on extant interactive video projects should focus on these factors, which are in all likelihood critical for positive learner response.

A systematic research agenda concerned with this composite of issues should provide the necessary information for designing interactive video systems that realize their educational potential.

Notes

1 David Cohen, "Educational Technology and School Organization," in *Technology and Education in 2020*, ed. Raymond S. Nickerson and Philip P. Zodhigtes (Hillsdale, N.J.: Lawrence Erlbaum, in press).

2 Kathleen Wilson, *Palenque: An Interactive Audio/Video Research Prototype* (New York: Bank Street College of Education, 1986, 1987).

3 Samuel Gibbon, "The Electronic Learning Environment of the Future," in *The Future of Electronic Learning*, ed. Mary Alice White (Hillsdale, N.J.: Lawrence Erlbaum, 1983), p. 4.

4 Lemuel Schaffer and Michael Hannafin, "The Effects of Progressive Interactivity on Learning from Interactive Video," *ECTJ* 34, no. 2 (Summer 1986): 89-96; and Michael Hannafin, "Empirical Issues in the Study of Computer-Assisted Interactive Videos," *ECTJ* 33, no. 4 (Winter 1985): 235-47.

5 Bank Street College of Education, Center for Children and Technology, *Palenque: An Interactive Audio/Video Research Prototype* and *The Voyage of the Mini Season I. Disc* (New York: Bank Street College of Education, 1987); Project Athena Visual Courseware Laboratory (Cambridge: MIT, 1987); and WNET-TV New York, Interactive Technologies Learning Laboratory, "Beyond Einstein: A Case Study in Interactive Television," (New York: WNET-TV, 1987).

6 D. R. Anderson and E. P. Lorch, "Looking at TV: Action or Reaction?" in *Children's Understanding of Television*, ed. J. Bryant and D. R. Anderson (New York: Academic Press, 1983).

7 Tom Synder and Jane Palmer, *In Search of the Most Amazing Thing: Children, Education, and Computers* (Reading, Mass.: Addison-Wesley, 1986), p. 84.

8 On MIT's Project Athena, see Douglas Morgenstern, "Simulation, Interactive Fiction and Language Learning: Aspects of the M.I.T. Project"; idem, "The Athena Language Learning Project: Computers in Research and Teaching," *Hispania*, September 1986, pp. 740-45; and Russell Gant, "Visual Courseware: Learning French, in Paris, in a Movie," *Optical Insights* 1, no. 1 (January/February 1987): 8-15.

9 Jerry A. Fodor, Thomas Bever, and Merrill Garrett, *Psychology of Language* (New York: McGraw-Hill, 1974).

10 Jean Jacques Rousseau, *Emile*, trans. Alan Bloom (New York: Basic Books, 1979).

Developing Thinking Skills with Computers

JOHN B. BLACK, KAREN SWAN, and DANIEL L. SCHWARTZ
Teachers College, Columbia University

Computers, in and out of education, are powerful retrievers of information. In an age of information, our key need is not to retrieve more of it with yet greater ease, but to reason more astutely. Will working with computers and new logic-oriented languages help?

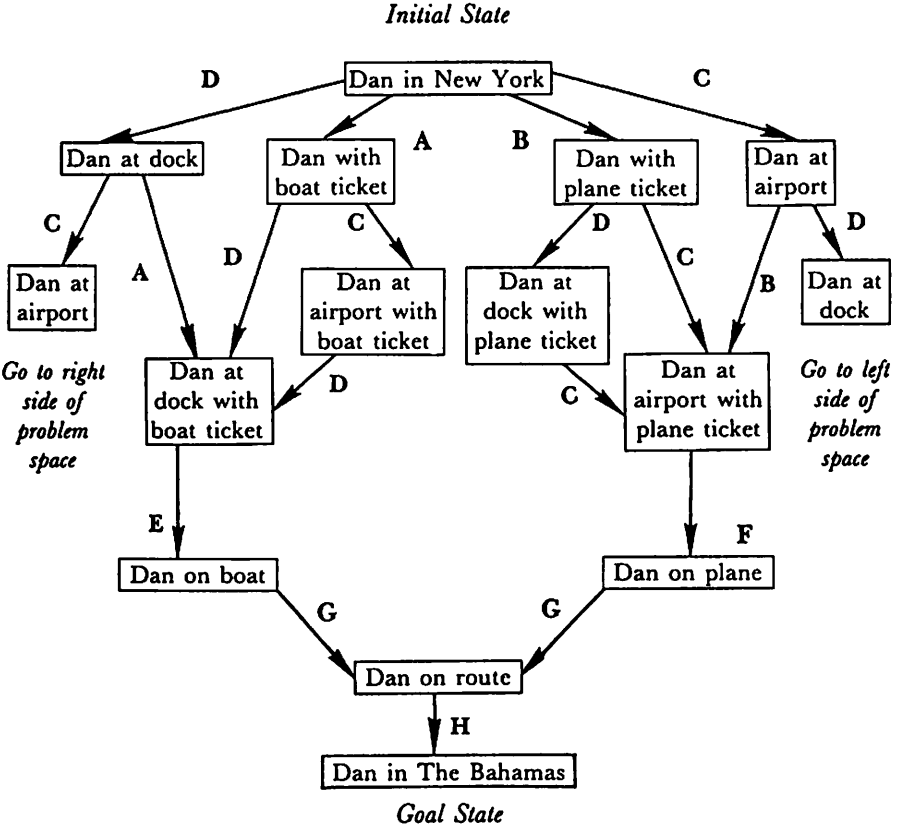
A number of recent surveys of students' skills in the United States have converged on a single, general conclusion: Students have mastered mechanical, lower-level skills in a variety of areas, but they are sorely lacking in the ability to do higher-order thinking in these areas. For example, the National Assessment of Educational Progress (NAEP) found that the problems with student writing were not in mechanics but in thinking and organization.[1] Another NAEP study found that young adults could read a short newspaper story to find a specific fact, but could not summarize an argument presented in an editorial.[2] Yet another NAEP study found that students' computational skills were fine, but that their problem-solving abilities were weak.[3] Thus we must find better ways of teaching higher-order thinking skills to students in all domains, because the current approaches are not working. In this article we argue that computers can be used effectively to convey thinking skills to students.[4] Specifically, we describe how to use Logo programming to teach problem-solving skills in a way that will generalize over specific domains, and we discuss how to use Prolog programming to teach generalizable reasoning skills. We also provide a brief discussion of using computers to teach two kinds of reasoning that critics have accused computers of undermining—reasoning with images and reasoning from experience.

Before claiming that we are teaching and measuring problem-solving or reasoning skills we should have a clear view of each. Problem solving, for example, is not a single skill.[5] Each given problem may require a different skill or combination of skills for its solution. Problem-solving behavior might be viewed as the selection and application of an appropriate strategy for a given problem. At other times, problem solving takes on the character of an "ah ha!" when the solution comes to mind after a good sleep. To instruct in problem solving we need to choose an amenable collection of strategies with

which to work. Therefore, we have chosen a few specific genres of problem solving to emphasize: subgoal formation, forward-chaining, backward-chaining, systematic trial and error, alternative problem representation, and analogical reasoning. One must also distinguish reasoning from problem solving and suggest the component skills that make up reasoning activity. By making a clear specification of the cognitive skills we plan to teach, both instructional objectives and methods of evaluation become feasible.

The notice of a *problem space* introduced by Newell and Simon[6] provides an organizing framework for discussing thinking skills. A problem space is composed of two general conceptual categories: states and operators. A state is a possible state of the world or problem situation (represented by circles in Figure 1). An operator is something that can be selectively applied to a state to change it to another state (represented by letters labeling arrows in Figure 1). For example, if someone is in a state of hunger, then eating something (an operator) may change the state to one of satiation. States and operators should be defined at a level of specificity and emphasis appropriate for the problem. For purposes of human action, the states and operator just described are about the right level. However, if we were trying to program a robot to mimic human eating habits, then we would have to use a much more detailed level of description (e.g., location in living room, go to kitchen, location in kitchen, go to refrigerator, and so forth). The selection of an appropriate level of detail and direction of emphasis is most closely aligned with the problem-solving activity of choosing alternative representations. If one represents the states and operators for solving a problem incorrectly, creating an alternative representation might be more conducive to solution. Experts in physics evidently spend a large proportion of their time correctly representing or formulating the problem state and the applicable operators before attempting to solve a problem. Novices, on the other hand, quickly choose a less appropriate representation of the problem state (generally a familiar equation).[7]

Given these two elements, states and operators, we can begin to construct a problem space. A problem space can be graphically imagined as an upside down tree. At the top of the tree is the initial state of affairs, for example: Dan is in New York (see Figure 1). At the bottom of the tree is the goal state: Dan is in the Bahamas. In between the initial state and the goal state we find a collection of intermediate states that can be derived by applying the operators to the immediately prior (above) state. From the initial problem state and given our limited set of available operators in the figure, Dan can reasonably apply four operators: Buy a boat ticket; Buy a plane ticket; Go to the airport; Go to the dock. Applying each of these operators yields a new state. Applying appropriate operators to the newly derived states gives us further states, until one (or more) of the newly created states is the goal state.



A Limited Set of Operators

A — Buy a boat ticket
B — Buy a plane ticket
C — Go to airport
D — Go to dock
E — Board boat
F — Board plane
G — Ride to Bahamas
H — Disembark

Figure 1. An Example Problem Space. Each box in the diagram represents a state. Each line represents a transformation into a new state according to the operation associated with the letter. Some possible paths have been omitted, such as the one that takes Dan from the airport to the dock. Reason enables us to find Dan's possible states. It also tells us which operators can be applied to any state. Dan cannot board a plane (F) unless he has bought a ticket (B) and gone to the airport (C). Problem-solving activity attempts to choose the best path from initial state to goal state (e.g., B –> C –> F –> G –> H).

There are two important points to be made. First, the problem space represents all of the possible states that can be derived using the operators. In reality, Dan will not reach all of the states. However, when reasoning about the problem of getting to the Bahamas, Dan should be able to reason out all these possibilities. Second, not all operators can be applied to all states. Dan cannot disembark before he has boarded a vessel.

With this introduction to the framework of problem spaces we can use it to make some distinctions between types of problem solving and deductive reasoning. Once one has solved the problem of selecting an initial representation of the problem and choosing the available operators, reasoning comes to the fore. For the current discussion, deductive reasoning is the ability to decide which operators may apply to a particular state and the ability to deduce all of the possible consequences from each operator's application. Note that this is a somewhat more general conception of reasoning than the usual restriction of reasoning to application to operators from formal logic. Isolating logic problems from others seems artificial, so we will use this more general conception of reasoning.[8] From this perspective, reasoning is what enables us to create the problem space. With this ability to create the problem space of possibilities, problem-solving abilities allow us to navigate the space so that we can reach the goal. For example, reasoning says that Dan can go from the dock to the airport, the airport to the dock. Problem solving tells us that this is not likely to be an effective strategy for reaching his goal. A different path through the problem space is more effective.

This example of problem spaces does not give one a sense of the power of this formulation for explaining thinking. Let us imagine a more complicated problem space. For example, take the problem of solving the following algebraic expression:

Initial State: $X + 1 = X^2 + X - 3$.

Our goal state, or in this case goal states, should give us:

Goal States: $X = 2 \qquad X = -2$.

What would the problem space in between be like? It is immense. For the first transformations of the initial state we can derive a substantial range of intermediate steps, including nonproductive ones like

Intermediate State: $-X^2 + X + 1 = -X^2 + X^2 + X - 3$. Reason, as the application of rules to produce legal consequences, allows us to come up with such an expression. Reasoning with the rules of algebra prevents us from creating an expression like

Ill-reasoned State: $X + 1 > X^2 + X - 3$.

Using reason to generate all the possible states from the initial state and then the next intermediate states and so on creates a problem space too large to work with. We cannot keep track of all the possibilities simultaneously. Prob-

lem solving, in our usage, helps us choose the most promising path to the solution. Often we choose the wrong path. We might use the first transformation shown above, which leads us away from the goal. Expert problem solvers might choose a second state like

An Expert's First Intermediate State: $1 = X^2 - 3$,

which they know will lead more efficiently to the goal. How do experts know how to move through the problem space in this direction? No doubt a great deal of experience in the domain has taught them that this is a factoring problem that should get into the form of 0 = <*expression*>. This is an example of subgoal formation. Rather than looking way ahead to the final goal, the problem solvers set up an intermediate goal that they know will facilitate finding the final solution. Another approach might be to follow a particular path mentally for several steps to see if something promising develops. For example, one might follow a path that collects all the Xs on one side. This is an example of forward-chaining. Problem solvers mentally follow a chain of reasoning to see if it leads to a propitious state. If it does they will commit to following this chain and probably begin to take actions according to this line of reasoning (e.g., write down the transformations on paper). If it is not a promising approach, they can try another chain of deductions. Or they might start from the form of the solution to see if they can build a chain of reasoning that will lead them back to the initial state. This is backward-chaining. This is expecially effective if one has found the answer in the back of the book. They might create a new initial state $X = 2$. From this they will perform a series of operations to reach the original problem statement. On the papers to be turned in for grading, our problem solvers can then reverse the sequence of steps to show their proof for the solution. Another approach to the problem might involve "pruning" the problem tree. Through systematic trial and error, problem solvers can exclude whole lines of reasoning early on by recognizing that a top-level state cannot possibly lead to the desired goal. The last area of problem solving we consider is analogical reasoning. If students had seen several problems like this before, they could analogize that this problem is the same and use the steps that helped in the previous problems. It should be clear that expertise in algebra will considerably aid problem solvers in creating the problem space according to the principles and axioms of algebra. Similarly, domain knowledge of algebra will help problem solvers choose a correct strategy for solving the problem. Nonetheless, the strategies and reasoning skills used are found in all domains. Subgoal formation might be used by Dan to realize that he needs to reach the subgoal of being at the airport with a ticket. The critical question is whether the practice of these problem-solving skills in algebra would show some positive side effects in areas of knowledge like geometry, science, and even English. More to the point, will programming a computer provide a superior

medium for the development and extension of these abilities? With a clear demarcation of the general skills, we have good objectives to teach to and a clear eye toward the manifestation these skills will take in other areas of knowledge so that we can teach for transfer.

USING LOGO TO TEACH PROBLEM SOLVING

Because it uniquely combines concrete, formal, and procedural representations of ideas, computer programming is often prescribed as fertile ground for the teaching and learning of problem solving. The Logo programming language, in particular, was developed for just such a purpose.[9] Research to date, however, has failed to clearly identify either the cognitive mechanisms or the pedagogical approaches that cultivate the development of problem-solving skills *within* programming contexts, let alone the *transfer* of problem-solving skills from programming to other educational contexts.[10]

Failure to document the transfer of thinking and problem-solving skills is, of course, not at all new to educational research; it dates from Thorndike's early debunking of the notion that the study of Latin improves reasoning.[11] It would seem that either we have yet to find a way to teach problem solving in our schools, or that we have yet to find a way to measure its transfer from specific to more general academic domains.

We will summarize here an initial study investigating the transfer issue in its contemporary programming/problem-solving incarnation.[12] We sought a finer-grained analysis of the problem-solving strategies involved in Logo programming environments, of the pedagogical approaches that might help cultivate such processes, and of the cognitive mechanisms involved in the transfer of these to noncomputer contexts. Several notions guided our inquiry.

To begin with, the teaching and learning of Logo has been notoriously nondirective.[13] The long history of failure to discover transfer effects suggests that the transfer of problem-solving skills is hardly automatic. We determined, therefore, to focus attention on the problem solving in programming. We devised introductory, off-computer activities designed to highlight particular problem-solving techniques, and followed these with sets of programming problems to which such strategies were particularly amenable. At the same time, we tried to remain faithful to the spirit of the educational philosophy espoused by Logo's creators.[14] We made the introductory activities both concrete and syntonic; we followed an open-ended, project-oriented approach in the problem sets; and we maintained a Socratic, guided-discovery pedagogy in our interactions with students.

Second, we believe that problem-solving abilities are themselves instances of a larger ability to think logically and abstractly, and that this ability, which Piaget terms *formal operational*,[15] develops slowly during the period

when a child is in middle and junior high school. We accordingly addressed our research toward children in this age range in an attempt to determine whether and/or when Logo programming/problem-solving experience might be useful.

A third consideration involved the generality of the concept of *problem-solving abilities.* Many authors have decomposed this into a number of distinct skills or strategies.[16] Certain such strategies seem more applicable to programming problems in general, children's programming in particular.[17] It seems probable, moreover, that of these, differing strategies become available at differing points in a child's development.[18] We identified six problem-solving strategies that we believed might be useful to children programming computers at some point in their development. These were subgoals formation, forward-chaining, backward-chaining, systematic trial and error, alternative problem representation, and analogical reasoning. In an effort to isolate the particular techniques most relevant in programming domains and/or most available for transfer to problem solving in other domains, we developed our instructional units and our measuring instruments around these six strategies.

Another consideration in the development of problem-solving skills involves the metacognitive monitoring of both the abstraction of such skills from particular contexts and their application in new domains.[19] There is some reason to believe that the explicit modeling of metacognitive behaviors helps students isolate and assimilate them.[20] Our use of introductory activities to highlight the specific strategies being explored and our adherence to a guided-discovery pedagogy reflect a concern with explicitly invoking metacognitive behaviors. In addition, we devised worksheets for students to fill out, asking them to delineate the problem space by identifying, in words and/or pictures, the *givens, goal,* and *solution steps* involved in each problem solution. We hoped that such deliberately forced attention to what Polya calls "understanding the problem"[21] would develop a corresponding metacognitive habit.

Finally, the teaching and learning of Logo have been, for the most part, restricted to the *turtle graphics* domain generally associated with the language. Indeed, the major Logo-problem solving studies appear to be based solely on graphics programming.[22] The Logo language, however, is far richer than turtle graphics alone would suggest. Logo makes *list manipulation* accessible to young children.[23] Programming with lists invokes a context quite different from graphics, a context more closely representing the workings of the language itself—hence, perhaps, more suited to problem solving within it. Additionally, research by Gick and Holyoak suggests that problem solutions are more easily abstracted and generalized from multiple, as opposed to single, base domains—that the transfer of problem solutions more readily occurs when these are initially encountered in more than one context.[24] We thought

that students working on problems in both graphics and list-manipulation contexts might be more likely to abstract and apply the problem-solving strategies they were using in new domains. Seeking to distinguish between varying base domains by varying the contexts of the problem sets, we created three student groupings and gave one group solely graphics problems to work on, one group solely list-manipulation problems, and the remaining group both graphics and lists problems.

Our subjects were 133 students in the fourth through eighth grades of a private suburban elementary school, all of whom had previous experience with both graphics and lists programming in Logo. They were randomly assigned by grade to one of the three contextual groupings. These remained constant across six instructional units corresponding to the six identified problem-solving strategies. A consistent instructional sequence was followed for each strategy unit. Our experimental design can be conceived, then, as a *problem-solving strategies* by *contextual groupings* by *grade levels* matrix, with context and grade-level groupings being between-subjects variables, and strategies, a within-subjects variable. The dependent variable was pre- and post-test scores on six measures of facility with particular problem-solving strategies.

Students were introduced to each problem-solving strategy through a whole-group activity designed to provide a concrete, off-computer model of the processes involved in it. For example, forward-chaining strategies were introduced with a treasure hunting game in which students followed a sequence of clues to discover a hidden treasure. For backward-chaining, we reversed the game and had student groups create the treasure hunts, an activity that has to be done from the treasure back to the first clue. Concrete strategy activities were followed with noncomputer example problems that were again presented to the whole class. Example problems were classic puzzles, such as the missionaries and the cannibals, the water jugs, and the heavy coin problems (See Table 1).

The group work of the introductory exercises was followed by individual or paired work on the problem-solving sets. Students were required to write solutions for three programming problems, all of which were particularly amenable to solutions employing the specific problem-solving strategy currently being explored. The first two core problems varied according to student groupings: Students in the graphics condition were given two problems involving graphics manipulation, students in the lists condition worked on two list-manipulation programs, and students in the two-domain condition had one graphics and one lists problem to solve. The third programming problem in each problem set served as a test of students' facility with each particular strategy across groupings. These problems were the same for all students and involved arithmetic problems, a third and unfamiliar programming context.

Table 1. Example Problems from the Logo Study

The missionaries and the cannibals. Three missionaries and three cannibals are on one side of a river with a boat that will hold only two people at a time. They want to cross. The problem is that if the cannibals outnumber the missionaries at any time during the crossing, they will eat them. How can everyone get safely across?

The water jugs. Given a jug that holds exactly seven quarts of water, a jug that holds exactly three quarts of water, and an infinite water supply, how can you empty and fill the jugs to end up with exactly five quarts of water?

The heavy coin. Given a pile of identical-looking coins, twenty-three of which are of equal weight, while one is heavier than the rest, and a balance scale with which the weights of any two piles can be compared, find the heavy coin in the least number of weightings. (It can be done in three.)

Students worked on the problems during two forty-five minute class periods each week. A teacher and an intern were available for help with all the problems during alternating class periods. Both maintained a guided-discovery approach toward student assistance, attempting to lead students to problem solutions, rather than telling them outright. For each problem, students were asked to fill out a problem-solving worksheet on which they showed, in words and diagrams, the givens, the goal, and the transformation steps in their problem solutions. In addition, they were required to turn in a listing and a run of their programs.

Instructional treatment lasted approximately three months. Subjects were tested both immediately before and after the entire six-unit intervention for their ability to correctly solve problems necessitating the use of each of the six problem-solving strategies. Five of the six instruments were devised by us. We attempted to make these specialized tests as independent of verbal aptitude as possible. Thus, for example, the test for forward-chaining ability was a pencil-and-paper derivative of the computer game "Rocky's Boots";[25] the analogies test contained visual as well as verbal analogies. For the sixth measure, that of students' ability to generate alternative problem representations, we used the figures subtests of the Torrance Test of Creative Thinking.[26] Different but analogous problems were given on the pre- and post-tests for each problem-solving technique, and differences between the two were examined using analysis of variance. The results surprised us.

On the one hand, we anticipated significant differences between pre- and post-test scores among contextual groupings. None materialized. On the other hand, previous failures to discover transfer effects gave us little reason to believe we would achieve significant overall differences, but, in fact,

Table 2. Results of the Logo Study

Subgoals		*Forward-Chaining*		*Backward-Chaining*		*Systematic Trial and Error*		*Alternative Representations*		*Analogies*	
Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post
7.5	9.8	8.4	10.4	10.9	11.2	5.1	9.5	95.8	135.7	18.4	19.5

There were essentially no differences in the contextual groupings (i.e., lists context, graphics context, or two contexts), so the results are collapsed across the contexts in this table. The numbers represent the scores on pretests before training in problem solving with Logo and on comparable post-tests after the training. There are significant differences between the pre-test and post-test scores in all cases except backward-chaining.

highly significant differences were found for all strategy groupings except backward-chaining (see Table 2). In particular, improvements in performance were found for subgoals, forward-chaining, systematic trial and error, alternative representations, and analogies. Such findings seem to demonstrate the transfer of these strategy skills from Logo to noncomputer domains. It is noteworthy that our finding of significant increases on the Torrance tests replicates, with an older student population, Clements and Gullo's earlier promising results.[27] Taken together with the lack of results concerning contextual groupings, however, they are not as discriminating as we would like, hence, not transparently explicable.

We had anticipated significant differences among varying contextual groupings because we believed that students solving problems involving list manipulation would gain a clearer understanding of the deep structure of the Logo language and so would show greater improvement in test scores than students working solely on graphics problems. Moreover, we thought it likely that experience with both graphics and lists would be more likely to facilitate the generalization of individual strategies then either experience alone, and so would produce the greatest increases in test scores. Neither hypothesis is supported by the findings. It may be that all students' previous experience with list processing was enough to develop a deep structural understanding of the language, or that two problems were not enough to make a discernible difference. Or it could be that the arithmetic problems, intended as a check on equivalent learning, effectively served instead as another problem-solving context, thus providing all groupings with multiple-domain experience. Future research will concentrate on better delimiting each of these variables in the hope of discovering the specific cognitive processes involved in transfer.

For the present, therefore, we are left with pedagogical explanations for the transfer of the particular problem-solving strategies we observed. We believe these would include the deliberate, forced attention to well-defined problem-solving ideas, the explicit modeling of the metacognitive behaviors

involved in problem solution, the required working through and corresponding proceduralization of problem solutions, and the attempt to acquaint students with the deep structure of the problem domain and of specific problem-solving techniques. In a recent article, Salomon and Perkins write: "Transfer . . . benefits from a high teacher-student ratio, Socratic interaction with the learners, great sensitivity on the part of the teacher for the ebb and flow of enthusiasm and understanding in the individual student, calculated provocation of abstraction and connection-making, and so on."[28] It could well be that the transfer we found results from similar features in our intervention. Most salient among these, and what may distinguish the instruction in our study from that of other such investigations, is the forced focus on problem-solving strategies and their procedural application in solving Logo programming problems.

On the other hand, the feature that categorically distinguishes our intervention from those reported in other studies is that our students were as familiar with list manipulation as they were with turtle manipulation. It is our contention that experience with lists yields a far clearer understanding of the deep structure of the Logo language and that such understanding facilitates problem solution within it. We would further argue that unless problems are solved with some nontrivial degree of understanding, there can be no transfer of problem solutions to other domains. Thus, even though we found no distinctions among contextual groupings supporting our contention, we believe it possible that our students' familiarity with list manipulation could have provided them with the critical domain knowledge necessary for understanding the particular problem-solving strategies being explored. Overall nondiscriminatory results for the test of backward-chaining ability may support such claims, in that backward-chaining is typically a novice strategy. Forward chaining is more an expert technique, based on a greater degree of domain knowledge.[29] That all ages of students exhibited similar increases in this particular strategy (and no other) further supports such argument.

Significant differences between pre- and post-test scores among grade levels were anticipated and were found for the subgoals-formation, systematic-trial-and-error, alternative-representations, and analogies strategies. In addition, the figures reveal a correlation between test scores and grade levels for all tests except backward-chaining, indicating that our tests were in some sense discriminating, and that they were discriminating in the direction that would be expected. Moreover, differences between scores among grade levels are, in general, less than pre-/post-test differences, indicating that real changes in student abilities took place as a result of the instructional intervention.

The findings of differential pre-/post-test differences among grade levels indicates developmental differences in the usefulness of particular problem-solving strategies. Specifically, we found greater increases on tests of subgoals-formation and systematic-trial-and-error strategies among younger

students. These results suggest that students in the middle school grades are optimally ready to acquire these strategies. This finding is particularly interesting in that Piaget argues that it is facility with precisely these skills that indicates formal-operational skills.[30] He places the onset of the formal-operational stage at twelve years of age, which falls right at the end of middle school. Correspondingly, we found greater increases among older students on tests of their abilities to generate alternative representations and appropriate analogies, which suggests that *readiness* to acquire these skills occurs later in a person's development. The implications of such results for the teaching and learning of problem solving are wide-ranging. In our opinion, further investigations focused on these particular variables could prove most fruitful.

USING PROLOG TO TEACH REASONING

Thus far we have touched on the position of reasoning in determining possibilities within a problem space. This has been useful for distinguishing and explaining problem solving. At this point it seems worthwhile to look more closely at the elements of analytic reasoning in developmental terms. Perhaps the most prominent feature of analytic reasoning is its generality. The same set of reasoning tools can be found in a variety of areas ranging from science to syntax. This generality is gained through the high level of abstraction and symbolism in reasoning activity. In fact, its formal representation, predicate logic, allows one to reason without any reference to the real world. According to Piaget, abstract reasoning abilities develop in the formal-operational stage, around middle school age.[31] Although there is substantial debate about the validity of positing a formal-operational stage, some of the hallmark abilities of this alleged stage of development do show up consistently in mature thinking.[32] Using some of these abilities as starting points in discussion, we can at least suggest the broad targets of our research. Furthermore, attention to developmental issues might uncover optimal ages for teaching different aspects of reasoning, even if they do not fall along the lines of Piagetian developmental stages. A few salient reasoning abilities that emerge with mature reasoning are the tendency and ability to determine all the possible states of affairs and then systematically discover which is the real solution (i.e., creating a problem space and pruning it with systematic trial and error); utilizing the hypothetico-deductive method of variable isolation; using reversibility to justify a solution; and, evaluating interpropositional expressions. The hypothetico-deductive method can be understood within the context of problem spaces. Scientists hypothesize that some variable may be causing a particular phenomenon. Then they deduce what the consequences must be if the hypothesized variable is the agent. Creating a problem space from the logically predictable effects of the variable, they go to nature to confirm or deny their hypothesis. They match the logically neces-

sary results of the hypothesis with the empirical results. The hypothesis is emphasized prior to the facts.

Reversibility can also be elucidated in the context of the problem space. Reversibility is often found in mature thinking when justifying the famous Piagetian conservation tasks.[33] For example, a child might justify why the short, fat glass has as much water as the tall, narrow glass: "The fat glass must have the same amount of water as the tall glass because if I poured the water from the fat glass into a tall glass it would be the same height as the other tall glass." Reversibility is what allows one to use backward-chaining, to move from goal to initial state and inversely from initial state to goal. Interpropositional evaluations focus on judgments about the truth value of the rules or operations of a problem space. A concrete operational child can evaluate an expression like "Gertrude is in the blue house; Gertrude is not outside." However, only an abstract reasoner could see that "if Gertrude is in the blue house, she is not outside" is true regardless of whether Gertrude and the blue house actually exist. An abstract reasoner is able to judge the validity of a statement without reference to some empirical fact.

Although most people show formal operations in some areas of their life, many do not apply the rules of reasoning well in other areas. The less than peak performance of humans while reasoning is well documented. Miller and Cantor, in their review of *Human Inference* by Nisbett and Ross, succinctly capture the problems:

> If the authors' purpose had been to belittle human reasoning, they would have had much ammunition available they did not use. Years of research on reasoning and problem solving have uncovered many glaring weaknesses . . . preference for positive instances; neglect of alternative hypotheses; overreliance on familiar content; functional fixedness; the treatment of conditionals as biconditionals and the pervasive fallacy of asserting the consequent; the tendency to delete, change or add premises; and so on.[34]

These failures show in typical school activities: the inability to draw conclusions and inference from reading, the difficulty in understanding the importance of a control in an experiment, and the lack of critical evaluation of political rhetoric.

There is evidently room and a need to investigate whether reasoning skills can be taught using sound pedagogical principles and long-term exposure to the material. More specifically, it is worthwhile to see whether programming a computer is particularly well-suited. Beyond the usual claims for a programming environment,[35] Prolog appears to be ideal for instruction in reasoning. Teaching problem solving using Pascal or Logo involves the student in learning these languages, which may not bear directly on problem solving. Learning the correct syntax and sequence for creating a looping procedure

is not intimately associated with solving classes of everyday problems. Problem solving must be specifically taught. Prolog, in contrast, is a language in which the coding of the program takes the form of logical reasoning. Learning the features and techniques of Prolog is learning the rules of reasoning. With a strong claim like this, it is worthwhile to introduce Prolog and see if there might be any truth to the assertion.

Prolog represents an important alternative to more traditional programming languages like Fortran and Pascal. One might term these latter languages "procedural" languages. They act as linguistic interfaces between one's thinking and the instruction set and architecture of the machine. One thinks of how to solve a problem and a coder (perhaps oneself) decides how to program the most efficient solution, providing a set and sequence of procedures for the computer to use in manipulating the data. The translation from thought into something digestible for the computer can be a tedious and language-specific chore. Generally, instruction in programming is designed to help the student learn to think within the computer's world, lessening the gap between solution and execution.[36] Prolog, however, provides an interface between conceived solution and computer implementation that favors our natural reasoning and knowledge representations over the computer's instruction set. In the following paragraphs, we will offer a brief introduction to Prolog as a "thinking" or "relational" language as opposed to the aforementioned procedural languages.

Prolog is one implementation of the abstract, machine-independent formulation of computational logic called "logic programming." Logic programming was developed by Robert Kolwalski in the early 1970s, building on computational mathematical logic. Unlike procedural programming, logic programming requires only the declaration of facts and rules in a suitable logical formalism. The "programmers" of this abstract machine need not concern themselves with the operations of the computer but rather with the interaction of the knowledge embodied in rules and facts to derive new rules and facts. An example borrowed from Walker et al.[37] might prove illustrative. Let us assume that we have four cities connected by three roads, the initial state of the problem space. This could be stated in a predicate formalism as:

road(city1, city2) road(city 2, city3) road(city3, city4)

meaning that there is a road connecting each of the city pairs. Our concern is whether there is a route between two cities. For example, is there a route from city1 to city3? We need to come up with a relationship between roads and routes that will capture our knowledge of them. A first step toward representing this knowledge might be done with the logical relation of implication (modus ponens):

route(X, Z) <— road(X, Z).

This states that there is a route between two locations (represented by the variables X and Z) if there is a road between these two locations. This first rule is inadequate for showing that there is a route between city1 and city3 as there is no single road connecting them. What we need is a second rule, which explains that there is a route between two cities if there are roads connecting the intervening cities:

route(X, Z) <– road(X, Y) & route(Y, Z).

This states that there is a route between two cities if there is a road between the first city and an intervening city and there is a route between the intervening city and the second city. Given this purely declarative description of the initial state and the rules or operators that capture our knowledge of the relation between routes and roads, logic programming will yield all of the following deductions, the full problem space:

road(city1, city2)	road(city2, city3)	road(city3, city4)
route(city1, city2)	route(city2, city3)	route(city3, city4)
route(city1, city3)	route(city2, city4)	route(city1, city4)

Logic programming describes a hypothetical language in which all the possible deductions from a given set of assertions can be found with the programmer providing only the given facts and available operators. Programmers need only be worried about the creation of the knowledge and interactions. They do not need to worry about the techniques for implementation. In contrast, procedural languages require one to specify a procedure for making these deductions. This requires an explicit creation of data storage locations, loops for iterating the possibilities, and conditional choices for exiting loops. There are inherent difficulties in capturing the theoretical simplicity and elegance of logic programming in a computing machine. In particular, the ability to create the full problem space means that a computer implementation would be terribly slow if it made all the possible deductions and then searched for the goal state within this space. A tradeoff between the efficient specificity of procedural languages and the elegant generality of a logic programming is a knotty problem. In 1973 Alain Colmerauer at Marseilles tried to capture with Prolog the desired generality of Kolwalski's logic programming while maintaining some degree of efficiency. Prolog has a variety of heuristics for optimizing solutions that move it away from the proposed generality of logic programming. There are constraints on the combinations of assertions one may use in Prolog and there are distinct programming techniques for ensuring a speedy solution to a problem. Short for "Programming in Logic," Prolog leans toward the programming aspect of solving problems. Fortunately, for the class of mini-problems in which most students will be engaged, there is no practical difference between logic programming and programming in logic. Students will not need to worry about Prolog programming constraints and the uses of programming constructs like the *cut* and *fail* statements.

In Prolog the steps for finding possible deductions are partially implemented through an "invisible" *unification algorithm* that matches assertions filling in or binding variables in one assertion to the facts of another. To demonstrate this we will introduce the two "goal" primitives of Prolog: IS and WHICH. IS determines if an assertion is true and returns the fact that made it true. WHICH tries to find all the possible situations in which an assertion is true. For example, a query like IS{X: road(city1, X)} will return X = city2. The unification algorithm will try to match the query road(city1, X) with some assertion in the program. It will find that road(city1, city2) matches and the variable X will be bound to the value city2. Here is a general delineation of the unification in a more complicated query:

IS{route(city1, city3)}

1. Try to match route(city1, city3) with an assertion in the program:
   route(city1, city3) MATCHES route (X Z) <—road(X Z)
   resultant bindings: X = city1, Z = city3.

2. To see if the head "route(X Z)" with which route(city1, city3) was unified is true it must see if its condition road(X Z) is true:
   X and Z are replaced with their bindings city1 and city 3 respectively.
   road(city1, city3) does not match any assertion.
   Thus, the assertion from step 1 is false. The bindings of X and Z are undone and the program looks for another assertion with which it can unify route(city1, city3).

3. Prolog will try to unify with the next available assertion:
   route(city1, city3) MATCHES route (X Z) <—
   road (X Y) & route (Y Z)
   resultant bindings: X = city1, Z = city3.

4. Prolog must now satisfy the conditions road(X Y) & route (Y Z) if the bindings from step 3 are to be considered valid.
   a. substituting X with city1
      road(city1, Y) MATCHES road (city1, city2)
      resultant bindings: Y = city2.
   b. substituting Y = city2 [Z is previously bound to city3]
      route(city2, city3) MATCHES route (X Z) <—road (X Z)}
      (1) As the assertion found in b has a condition that there must be a road between city2 and city3, this must be found true for route(city2 city3) to be true. Prolog will find this assertion in the data base. It will conclude that the condition for b is true. The condition for 4 must be true as both a and b are true. Finally, it will find the original query true as 3 must be true since 4 is true.

If one made a WHICH{X: route(city1, X)} query, Prolog would find all the instances of X that would satisfy the conditions for being considered a route.

Essentially, it would do the same process as above, remembering each binding that worked and returning the results:

X = city2 X = city3 X = city4.

The unification of variables with facts, the selection of rules to be tried, and the order of choosing the conditions to be checked is totally hidden from the programmer. This leaves programmers free to explore the knowledge and relationships required for finding the solution. Further, it allows thinkers to interact with the knowledge and assumptions they have used in trying to find the solutions for problems. Prolog's ability to deduce anything that is deducible means that if a program is unable to find a hypothesized solution, the programmers qua thinkers must review the knowledge they think characterizes the requirements for the solution.[38] As this knowledge is in a declarative and logical format, it is readily inspectable by natural reasoning methods. Errors found are errors in thought, not in program construction or improper syntax. Sterling and Shapiro summarize this nicely:

> We think that programming can be, and should be, part of the problem solving process itself; that thoughts should be organized as programs, so that the consequences of a complex set of assumptions can be investigated by "running" the assumption; that a conceptual solution to a problem should be developed hand-in-hand with a working program that demonstrates it and exposes its different aspects. . . . In contrast, formalizing one's problem and method of solution using the von Neumann instruction set [procedural programming languages] rarely has these beneficial effects.[39]

Prolog until recently received little attention in the United States. However, with its acceptance as the official language of the Japanese Fifth Generation Project in 1981, Prolog moved from an obscure language into the forefront of artificial-intelligence work. With its general theorem-proving ability, use of symbolic data, list-processing abilities, and efficient search methods, it is offering a viable alternative to McCarthy's function-oriented language LISP. More important for our concerns as educators, several appealing micro-computer versions are now available.[40] It is interesting to speculate about the role of Prolog with the recent development of parallel computers. The ability of parallel computers to do several operations simultaneously will require a dramatic change in typical procedural languages, which assume that operations occur sequentially. On the other hand, logic programming makes no such assumption, as it does not bother the programmer with problems of explicitly guiding the machine. Prolog can capture the power of this new generation of computers without changing the nature of the programming activity. With the advent of parallel machines, new programming logics will be developed that do not require the constraining heuristics of Prolog to work efficiently, yet they will share the commonality of being based on logic programming.[41]

We can now begin to describe how Prolog can be used to teach reasoning to students. The results of our current research project to evaluate empirically the use of Prolog to teach reasoning to middle-school students are not yet available, so we will illustrate this approach by presenting two examples of the instructional activities that are feasible for immediate use given little exposure to Prolog: reversibility and AND relationships (intersections) for defining membership. Although our instruction will use more elaborate interfaces built on the Prolog environment (e.g., a prebuilt data base, an expert system shell, a natural language front-end, and a front-end capable of graphically representing the knowledge relationships), the following exercises are easily implemented in any Prolog environment.

Reversibility is the ability to recognize that operations undo one another. The operations of subtraction and addition are an excellent example. As children develop a more sophisticated understanding of arithmetic they should come to learn the reversibility of these operations.[42] A reasonable objective would be that the child should be able to demonstrate several operations that undo one another. At the more specific level of mathematics, the child should be able to use subtraction to solve a problem written in the form of $3 + X = 9$.

Prolog, as does reasoning, uses reversibility extensively to solve problems. In fact, most Prolog environments provide only the two primitives PLUS and TIMES for integer arithmetic. Subtraction and division answers are found by asking the computer to find the inverse relationships implied by PLUS and TIMES. A teacher might show the students how to do addition problems in Prolog:

```
PLUS(2 3 X)
```

Prolog would find the value of the variable X to equal 5. The teacher might elicit suggestions or ask students how to do subtraction. Eventually, it should come out that subtraction can be found by moving the variable to an addend position:

```
PLUS(2 X 5)
```

would yield $X = 3$. Moving X to the first position would give $X = 2$. The teacher might then ask the students to try some division problems using TIMES.

Once these operations are mastered they can be used to elaborate on the role of interconstraining variables in an equation (an example of interpropositional evaluation). For example, TIMES(X Y 20) gives all the factor pairs of 20:

$X = 1,\ Y = 20$
$X = 2,\ Y = 10$
...
$X = 1,\ Y = 20.$

This can be elaborated on depending on the inclinations of the class to create a program that finds the greatest common factor of two numbers. For some Prolog versions, this requires a single implication:

GCF(X Y) <– TIMES(A B X) AND TIMES(C A Y) and PRINT(A).

The PRINT(A) will force Prolog to display the answer bound to A. Once this implication has been typed into the computer (according to the syntactical requirements of the system, usually trivially different from the above version), questions about greatest common factors can be answered. To find an answer one might query:

WHICH {GCF(14, 21)}.

To check a proposed solution one might enter:

IS{7 <–GCF(14, 21)}.

Students can explore the role of variable positions within the GCF implication as a general unguided exploratory activity.

An enrichment activity might consist of writing an implication that finds the least common denominator of two numbers:

LCD(X Y) <– TIMES(X A Z) AND TIMES(Y B Z) AND PRINT(Z).

Clearly these latter activities are of some difficulty to work through and depend tremendously on teacher and student competence for success. However, the initial lesson on reversibility should be simple and useful.

Another simple activity, introducing the notion of necessary attributes and intersections, would involve creating a data base and using the conjunction AND in queries. The first step is to settle on a formalism for entering facts into the data base. A simple one might use the two predicates HAS and IS-A, which will each accept two arguments:

HAS(Red-Hair Mary), IS-A(Female Mary).

Once this is decided on, teachers might have students enter facts about themselves into the computer. They might require that at least the students include statements about hair color, height, age, and so forth. One's imagination is the only limitation here. Once the computer users have entered their facts, the class's facts should be transferred via disk to one computer. Teachers will then ask if anybody would like to find out about something in the data base. Questions will probably take the form of IS{IS-A(Female Mary)}. These are factual questions. More abstract questions can take the form of WHICH{X : IS-A(X Mary)}, which will give all the IS-A information about Mary. Students might be engaged in a conversation that discusses the

different results if one asks WHICH{X : IS-A(Female X)}. This latter question will give all the females in the data base.

Once this initial grasp of the data base query and the use of variables is developed, students are ready to move on to the concept of conjunctive constraints (or necessary but singularly sufficient conditions) for class membership. As an example the teachers might say that they would like to make a club of all the children who have blonde hair and are left-handed. How might they query the data base to find out who could belong? One likely answer is to make a query that asks for all the left-handed people and then write these names on the board. Next make a query for all the blonde people and write these on the board. Then one could see who is in both groups. A discussion revealing the terms *necessary* and *sufficient* could ensue. That is, left-handedness and blondeness are necessary attributes but by themselves are not sufficient for belonging to the club. Using Venn Diagrams to help illustrate set membership, teachers could then introduce the use of AND in simplifying the query:

```
WHICH{X : HAS(Blonde-hair X)
          AND IS-A(Left-Hander X)}
```

Once the notion of an intersection or conjunctive relationship is firmly in hand, a class might try to create some predicates that define the members of a set:

```
BELONG-TO-TEACHERS-CLUB(X)  <-HAS(Blonde-hair X)
                             AND IS-A(Left-Hander X)
```

Students can then use the new predicate to simplify their queries. This begins to introduce the notion of a conditional; X is true if Y is true. Although the students are probably not ready to evaluate the implications of conditionals, this would serve as a good introduction to the utility and form of conditional expressions. A venturesome class might begin to build complicated definitions that build off of other implications:

```
BELONG-TO-MY-CLUB(X) <- BELONG-TO-TEACHERS-CLUB(X)
                        AND HAS(Money X).
```

These lessons attempt to give a flavor of the flexibility and relative ease of using Prolog for a variety of reasoning problems. With a thoughtful development of instruction, students could develop quite an impressive structure of reasoning and programming skills. Using basic reasoning constructs, children could eventually create interactive computer games by specifying the rules of play and the facts that represent the "game board." They might even be able to make the computer a good opponent in such a game. During all

the students' projects they will be learning and operating with highly abstract reasoning.

OTHER THINKING SKILLS

We have chosen to discuss a few thinking skills in some detail here in order to illustrate how computers can be used effectively to convey them to students. However, we do not mean to imply that only these kinds of thinking skills can be conveyed by computer. Critics of computer uses in education have argued that computers can be used only to teach the sorts of rule-oriented thinking skills we have emphasized here.[43] Few would argue about the value of these rule-oriented skills, but clearly they do not encompass all human thinking skills; to limit the curriculum to these alone would be a disservice to students. The capabilities of computers are not, however, limited to conveying general rule-based thinking, as a brief discussion of reasoning from images and from specific experiences—skills frequently cited as slighted in computer use in education—will show.

Suppose we want to teach students about how a steam engine works or how radio and television waves work. Understanding how such devices work involves heavy use of mental simulation utilizing the human imagery system.[44] For example, in order to understand what happens to radio and television waves in a big city with tall buildings, students have to use dynamic mental images to visualize waves spreading out in spherical wave fronts from an antenna, then parts of the waves reflecting off the tall buildings to set up an interference pattern. Similarly, with the steam engine, students need to be able to visualize what happens as various valves are opened to certain levels and water or steam flows through various pipes and chambers.

Students can gain some insight into steam engines by manipulating toy steam engines, but that alone cannot teach them how to visualize and simulate the internal workings of the engine. This mental simulation ability can be conveyed effectively using a computerized graphic simulation of how the steam engine works; in fact, the Navy uses such a computerized simulation to train boiler room personnel for Navy ships.[45] The radio and television wave example is even more compelling: Because radio and television waves are invisible, there is no way students can physically manipulate them to see what happens in the real world. A graphic computer simulation serves as a powerful way to convey the dynamic-imagery thinking abilities students need to reason about such wave behavior.

The research of Taylor and Cunniff provides another example of the use of graphic computer displays to convey imagistic thinking skills to students.[46] In particular, Taylor and Cunniff have shown that teaching students programming using a graphic programming language eliminates many misconceptions and allows the students to comprehend programs faster. Thus, far

from undermining human imagery skills as some critics have suggested,[47] the use of computers in education provides a more effective way of conveying how to think in images.

Another criticism of the use of computers in education is that it encourages students to believe that learning general rules is all that is involved in learning to think.[48] No one would want to be treated by medical school graduates who know all the general rules of medicine but have had no chance to supplement this general knowledge with the experience provided by a residency. In fact, this notion that a true expert reasons not just from general rules but also from specific experience is a powerful idea that should be taught to students. As the critics have charged, teaching students simple turtle graphics in Logo does not provide an environment in which to teach the value of reasoning from experience and may even discourage students from thinking that way. This point is a specific instance of a more general point made by McClintock—that educational computer systems to date are informationally impoverished, that is, they do not contain anywhere near the amount of information that is contained in books.[49] However, as McClintock goes on to point out, this is changing as more powerful computers with immense storage capacities become available. In fact, storage media like CD-ROM can store much more information than books (a single CD-ROM resembles a small library more than a book).

However, just providing large amounts of information is not enough; students have to be taught to reason with it effectively. We argue that the kinds of strategies for using computers in education described earlier can be used to teach students to utilize both rule-based and experience-based reasoning. Although Logo can be used for this purpose if the usual turtle graphics programming is supplemented with list processing, Prolog is particularly convenient for conveying this idea because of its clear delineation of general rules, specific rules, and specific facts. In particular, students focusing on creating general rules when programming in Prolog will quickly run into trouble (i.e., have awkward programs that do not function effectively) and realize that they need to add specific, context-sensitive rules and structured descriptions (using list structures) to the general rules they began with. While we have no illusions that these representations completely capture knowledge acquired with human experience, they do capture some of it and illustrate to students in a compelling and concrete fashion the need to coordinate rule-based and experience-based reasoning.

Notes

1 National Assessment of Educational Progress, *Reading, Thinking and Writing: Results from the 1979–1980 National Assessment of Reading and Literature*, Report No. 11-L-01, October 1981.

2 National Assessment of Educational Progress, *Literacy: Profiles of Young Adults*, Report No. 16-PL-02, 1986.

3 National Assessment of Educational Progress, *The Third National Mathematics Assessment: Results, Trends and Issues*, Report No. 13-MA-01 (Denver: Education Commission of the States, April 1983).

4 Of course, we are far from being the first to argue this, but we offer some specific approaches that are unique.

5 J. Greeno and H. Simon, *Problem Solving and Reasoning*, Technical Report No. UPITT/LRDC/ONR/APS-14, February 1984.

6 A. Newell and H. A. Simon, *Human Problem Solving* (Englewood Cliffs, N.J.: Prentice-Hall, 1972).

7 M. T. H. Chi, R. Glaser, and E. Rees, "Expertise in Problem Solving," in *Advances in the Psychology of Human Intelligence*, Vol. I, ed. R. J. Sternberg (Hillsdale, N.J.: Lawrence Erlbaum, 1982).

8 Greeno and Simon, *Problem Solving and Reasoning.*

9 See Seymour Papert, *Mindstorms: Children, Computers and Powerful Ideas* (New York: Basic Books, 1980).

10 See K. Ehrlich et al., *Issues and Problems in Studying Transfer Effects of Programming* (New Haven: Yale University Department of Computer Sciences, 1985); and Roy D. Pea and D. Midian Kurland, "Logo Programming and the Development of Planning Skills," in *Mirrors of Minds*, ed. K. Sheingold and R. Pea (Norwood, N.J.: Ablex Publishing, 1987).

11 See E. L. Thorndike, *Educational Psychology, Vol. 2* (New York: Bureau of Publications, Teachers College, Columbia University, 1913).

12 For a detailed account of this research, see K. Swan and J. B. Black, *Cross-contextual Transfer of Problem Solving Skills*, CCT Report 87-3 (New York: Teachers College, Columbia University, 1987).

13 See Uri Leron, "Logo Today: Vision and Reality," *The Computing Teacher*, February 1986, p. 26; Pea and Kurland, "Logo Programming and the Development of Planning Skills"; and Gavriel Salomon and D. N. Perkins, "Transfer of Cognitive Skills from Programming: When and How?" *Journal of Educational Computing Research* 3, no. 2, p. 149.

14 See Papert, *Mindstorms.*

15 See Jean Piaget, *Genetic Epistemology* (New York: W. W. Norton, 1971).

16 See G. Poyla, *How to Solve It* (Princeton, N.J.: Princeton University Press, 1973); W. A. Wickelgren, *How to Solve Problems* (San Francisco: W. H. Freeman, 1974); and Greeno and Simon, *Problem Solving and Reasoning.*

17 See Douglas H. Clements and Dominic F. Gullo, "Effects of Computer Programming on Young Children's Cognition," *Journal of Educational Psychology* 76, no. 5, p. 1051; Robert W. Lawler, *Computer Experience and Cognitive Development: A Child's Learning in a Computer Culture* (New York: Halsted Press, 1985); S. M. Carver and D. Klahr, "Assessing Children's Logo Debugging Skills with a Formal Model," *Journal of Educational Computer Research* 2, no. 4, p. 487; C. A. Clement et al., "Analogical Reasoning and Computer Programming," *Journal of Educational Computing Research* 2, no. 4, p. 473; and Douglas H. Clements, "Longitudinal Study of the Effects of Logo Programming on Cognitive Abilities and Achievement," *Journal of Educational Computing Research* 3, no. 2, p. 73.

18 See Piaget, *Genetic Epistemology*; Herbert Ginsburg and Sylvia Opper, *Piaget's Theory of Intellectual Development* (Englewood Cliffs, N.J.: Prentice-Hall, 1979); and Herbert Ginsburg, *Children's Arithmetic* (Austin, Tex.: Litton Educational Publishing, 1986).

19 See John H. Flavell, *Cognitive Development* (Englewood Cliffs, N.J.: Prentice-Hall, 1985).

20 See Carver and Klahr, "Assessing Children's Logo Debugging Skills with a Formal Model."

21 See Polya, *How to Solve It.*

22 See Seymour Papert et al., *Final Report of the Brookline Logo Project (Logo Memo 53)* (Cambridge, Mass.: MIT Artificial Intelligence Laboratory, 1979); Leron, "Logo Today: Vision or Reality"; Pea and Kurland, "Logo Programming and the Development of Planning Skills"; Clement et al., "Analogical Reasoning and Computer Programming"; and Clements, "Longitudinal Study of the Effects of Logo Programming on Cognitive Abilities and Achievement."

23 See Karen Swan, "Primarily Lists," *Pre-Proceedings of the 1986 International Logo Conference* (Cambridge, Mass., July 1986).

24 See M. L. Glick and K. J. Holyoak , "Schema Induction and Analogical Transfer," *Cognitive Transfer 15*, 1.

25 Learning Company, 1982.

26 E. P. Torrance, *Torrance Tests of Creative Thinking: Figural Tests* (Lexington, Mass.: Personal Press, 1972).

27 See Clements and Gullo, "Effects of Computer Programming on Young Children's Cognition."

28 See Salomon and Perkins, "Transfer of Cognitive Skills from Programming: When and How?" p. 164.

29 See Greeno and Simon, *Problem Solving and Reasoning.*

30 Piaget, *Genetic Epiustemology.*

31 For a readable discussion see Ginsburg and Opper, *Piaget's Theory of Intellectual Development.*

32 Flavell, *Cognitive Development.*

33 For an example of the role of reversibility in arithmetic, see Ginsburg, *Children's Arithmetic.*

34 G. A. Miller and N. Cantor, "Review of R. Nisbett and L. Ross, *Human Inference: Strategies and Shortcomings of Social Judgment*," *Social Cognition* 1 (1982): 83-93.

35 See Papert, *Mindstorms*, for an eloquent discussion of computer environments for learning.

36 L. Sterling and E. Shapiro, *The Art of Prolog* (Cambridge, Mass.: The MIT Press, 1986).

37 A. Walker et al., eds. *Knowledge Systems and Prolog* (Reading, Mass.: Addison-Wesley, 1987).

38 Sterling and Shapiro, *The Art of Prolog.*

39 Ibid.

40 *Turbo Prolog*, produced by Borland International (4585 Scotts Valley Drive, Scotts Valley, CA 95066) is a flashy new version of IBM PCs with 384K of RAM. It is appropriate for creating larger interactive programs as it is a compiled language with link-ups to Turbo Pascal and Turbo C. *micro-PROLOG*, produced by Logic Programming Associates, Ltd. (10 Burntwood Close, London SW18 3JU), is the original micro-computer implementation. Although not as visually attractive as Turbo Prolog, it is truer to the mainframe implementation of Prolog, is interpreted, is available for Apple II computers, and comes in a variety of interfaces for difference audiences. The *Simple* version of micro-PROLOG provides a pop-down menu to simplify interaction in the programming environment.

41 Sterling and Shapiro, *The Art of Prolog.*

42 Ginsburg, *Children's Arithmetic.*

43 For example, the papers in Douglas Sloan, ed. *The Computer in Education: A Critical Perspective* (New York: Teachers College Press, 1985).

44 J. B. Black and L. Marks. *Knowledge Acquisition Involving Dynamic Imagery*, CT Report 87-4 (New York: Teachers College, Columbia University, 1987).

45 J. D. Holland, E. L. Hutchins, and L. M. Weitzman, "Steamer: An Interactive Inspectable Simulation-based Training System," *AI Magazine* 5 (1986): 15-28.

46 Robert P. Taylor and Nancy Cunniff, "Moving Computing and Education beyond Rhetoric," *Teachers College Record* 89, no. 3 (Spring 1988): 360-72.

47 Douglas Sloan, "Introduction: On Raising Critical Questions about the Computer in Education," pp. 1-10; and H. L. Dreyfus and S. E. Dreyfus, "Putting Computers in Their Proper Place: Analysis versus Intuition in the Classroom," pp. 40-63, in Sloan, *The Computer in Education.*

48 Dreyfus and Dreyfus, "Putting Computers in Their Proper Place"; and John M. Broughton, "The Surrender of Control: Computer Literacy as Political Socialization of the Children," in Sloan, *The Computer in Education*, pp. 102-22.

49 Robert O. McClintock, "Into the Starting Gate: On Computing and the Curriculum," *Teachers College Record* 88 no. 2 (Winter 1986): 191-215.

Computer as Material: Messing About with Time

GEORGE FRANZ
The Computer School, New York City Board of Education
SEYMOUR PAPERT
Massachusetts Institute of Technology

Computers began in education with a charismatic aura that cannot remain characteristic of their long-term presence. As they become part of the everyday toolbox that kids can dig into, will they have any special value?

Computers, with their power and technological sophistication, fascinate just about everyone. Teachers, parents, administrators, and students agree that computers have added an important presence and dimension to educational settings. However, within the history of education, computers represent a very recent arrival on the scene and their role has not yet begun to be explored.

An examination of computer use in schools today reveals that students' interactions with computers are largely teacher-directed, workbook-oriented, for limited periods of time, and confined to learning about the machines themselves or about programming languages. Further, computers are located in separate labs and are not integrated into the standard curriculum. "Doing computer" in school is thought of as an exciting activity in and of itself. This separation is reflected in the often asked question: "Does what children learn with the computer transfer to other work?" The present separation of computers from other curricular areas is reflected too in arguments about whether computers might even be bad for children.[1]

The project described in this article approaches the computer in quite a diffferent sense. Instead of the familiar uses of the computer, which Robert Taylor has christened "Tutor, Tutee, Tool,"[2] the computers in this project are employed in a new way, which we call "Computer as Material."

The project reported in this chapter was carried out at The Computer School, New York City Board of Education, 100 West 77th Street, New York, NY 10024. The work presented here was aided by a grant from the Apple Education Foundation. Special thanks to Elizabeth Franz, Lillian Weber, Ted Chittenden, and Steve Shuller for guidance over the years and for assistance in preparation of this chapter.

The setting for the project was a junior high school science classroom in the New York City public schools. The classroom was well supplied with various materials from test tubes, pulleys, and microscopes to scrap wood, broken electronic devices, marbles, and the like. Also present in the room were computers and Logo. In this project, the students built devices for measuring time using any materials they wished. Some used string and a metal weight to make a pendulum, some used plastic containers to dribble sand—and some used computers. Our central focus is this use of the computer as just another type of material.

We mention one other closely related point of interest. The phrase "messing about" in our title is, of course, taken from a well-known paper by David Hawkins.[3] Marvelously entitled "Messing About in Science," it describes how he and Eleanor Duckworth introduced children to the study of pendulums by encouraging the students to "mess about" with them. This would have horrified teachers or administrators who measure the efficiency of educaton by how quickly students get to "know" the "right" answers. Hawkins, however, was interested in more than right answers. He had realized that the pendulum is a brilliant choice of an "object to think with," to use the language of Papert's *Mindstorms*,[4] one that can build a sense of science as inquiry, exploration, and investigation rather than as answers.

Just as pendulums, paints, clay, and so forth, can be "messed around with," so can computers. Many people associate computers with a rigid style of work, but this need not be the case. Just as a pencil drawing reflects each artist's individual intellectual style, so too does work on the computer.

THE PROJECT—TIMERS AND CLOCKS

Step one of the project was to bring the students to understand the need for the measurement of time. The teacher began by putting an empty glass jar over a lit candle and having the class watch the flame go out.[5] This was repeated several times, then the students' wristwatches were collected and the classroom clocks were disconnected.

Repeatedly the candle was lit and the jar was placed over it, and the students were asked to predict when the candle would go out. They quickly realized the need to develop a timing procedure—such as counting their heartbeats or breaths, or counting one-Mississippi two-Mississippi, and so forth. It might take fifty-seven heartbeats or fourteen breaths from the time the jar was placed over the candle until the flame was extinguished.

After perfecting their own "body timers," the students covered their eyes and raised their hands to indicate when they thought the candle had gone out. There was much discussion concerning different methods of timing and the accuracy of each, prompting the students to defend, evaluate, compare, and evolve their individual timing systems.

The need for something more objective than body clocks was sharpened by introducing environmental influences. Without any explanation, phonograph records (first a fast rock 'n' roll, later a slow Brahms) were played as the students carried out their timing methods. Further, students reentering the room on successive days were asked to predict from memory when the candle would extinguish. While some students' predictions remained quite accurate, most were significantly off. Discussion led to the conclusion that the rate of their body timers varied from one day to the next and that they were also affected by what went on in the class, such as the type of music that was played. It thus became obvious that the next step was to create a timer that was much more accurate and consistent from day to day.

What to do? How to proceed? Different suggestions were offered, most of them fantastic and impossible, often ideas centering around the creation of complex timers such as the gear-driven type on the wall or on their wrists. After a period of discussion, the teacher suggested that this was enough talk. "Let's get to work and make some clocks," was the challenge.

BUILDING CLOCKS

The room was well stocked with materials—in part because students were encouraged to bring in what a casual observer (and even the children) might call "junk," such as egg cartons, soda bottles, tin cans, and so forth. (The project would have been very different in a room full of only "sterile," store-bought equipment.)

The students set to work constructing timers, which took many different forms. Plastic cups taped together after having been filled with just the right amount of sand (determined experimentally) became crude egg timers. Water dripped out of a small hole in the bottom of a tin can and loudly plopped on a tin plate; the number of drops was carefully counted to tell elapsed time. A metal marble rolled through grooves chiseled out of pieces of wood; its speed, and therefore the time involved, could be controlled by varying the angle of the wood. Water slowly flowed into a cup on the up-end of a seesaw, and when the proper amount of time had elapsed, enough water filled the cup to make the seesaw tilt, at which point a piece of metal was tripped to complete a circuit that rang a bell. For students who found it hard to imagine a timer, there were library books with drawings, descriptions, and model plans of many different timing devices.

The room was also well supplied with material of a very different kind—computers and Logo—which some of the students chose to use in constructing their clocks. The first Logo timers were not really very different from the other clocks being built. Some examples included a regularly blinking screen, or a turtle alternately moving forward and then pausing, or the computer beeping at regular intervals.

While some students were speaking the language of Logo in order to achieve their goal of making a timer, others were speaking the language of a chisel, or of a battery and electric motor, or of a ball rolling down an inclined plane. Most of these languages were new to the children. What was important was that the students were learning to speak the languages of many different materials in the classroom in an attempt to create their clocks from ideas in their minds. When the students let their imaginations go, they found a variety of odds and ends for different explorations and investigations. The emphasis was on inquiry and learning, not on the type of material used. The computer was just one more material, alongside candles, crayons, ammeters, and rulers. The computer did, however, add dimensions not present in other materials, allowing students to go beyond the capabilities of the clocks constructed with the more commonly found materials.

BEYOND "MESSING AROUND"

While computers were used like wood, string, and electricity as material to mess about with, they evolved into something else as the Logo timers became more differentiated and sophisticated. The students began constructing Logo clocks that were highly accurate and precise, a goal not easily attained with the other materials. Many of the Logo clocks became as accurate as the students' wristwatches. These clocks ranged from a second hand that moved clockwise around the face of a square clock each minute to a digital readout of hours, minutes, and seconds. Some students added a beep for each second; others printed out on the computer monitor the number of seconds as they ticked by.

CALIBRATION: A FIRST CONNECTION TO MATH

The original problem of predicting when the candle would go out could be solved without using standardized units of time, that is, seconds, in the handmade clocks. As long as a clock beat with a fixed rhythm, it could be used to find out that the candle burned for, say 47 units, while another "slower" clock might have used only 27 units. Now a new question was posed to the class: "How do the units of our clocks compare with the standard unit of seconds?" This is the problem of calibration, and it gave rise to a new phase of the project. The goal this time was to relate the unit of their clocks to the standard time unit of seconds.

The precision of the Logo clocks did not come automatically from the precision of the computers themselves. Like the other clocks, these also needed to be calibrated. It is easy to write a Logo program that will repeat an action with a fixed period. However, calibration was still needed to make the period precisely match normal time units. The computer clocks were just like the

noncomputer clocks in this respect, so calibration is discussed in its general form.

Suppose you have a process that repeats about once a second and you want to adjust it to repeat *exactly* once a second. How would you proceed? Some of the students began by trying to adjust each individual unit to once per second. Discrepancies were easy to see when the intervals between their clocks and actual seconds were very far apart (say once per two seconds or twice per second), but when the intervals came close together, judgment could not be made by eye. Students tried using intermediate processes—for example, clicking their fingers in time with a watch to signal one beat per second and then comparing this with the period of their own clocks. Such tricks improved their estimation but were still rather limited.

A suggestion that spurred more fruitful directions of inquiry was the idea of thinking in terms of series of cycles rather than individual cycles. Instead of trying to time a single event that took one second, students could make a test run by timing twenty of these events—which should take twenty seconds. This was quite an improvement.

Students also incorporated the concept of averaging numbers to their data. They realized that in order to ensure the accuracy of their clocks, they had to make *several* test runs and then average the results of the runs. Obtaining an average had never been quite so effortless in math class! This was the first time any of the students had come across the concept of statistical averaging—using more than one trial run since it was possible that they had made a gross error on just one test try.

The connection with statistics was only one of many ways in which the work on clocks led into analytic reasoning. We saw another example in the Logo clocks. If the clocks used WAIT 20, the program counted out the seconds too slowly. (The Logo manual states that the command WAIT 20 pauses for one second.) Interestingly, when they analyzed the problem, the students often thought that their clocks ran too slowly because "twenty was too small," so they changed it to twenty-five. This made their timers pause for a longer period of time and therefore go even more slowly. Trial and error mingled with ample quantities of thought and discussion led them to realize that the smaller the number following WAIT, the shorter the wait. Then it was simply a matter of finding the right number by further trial and error. In the case of the noncomputer clocks, too, it was not always obvious to the students which direction of change would increase the period. Should one lengthen or shorten the pendulum's string or maybe increase the weight of the pendulum bob?

In the work on calibration, careful measurement revealed that a large graduated cylinder with a small funnel in its mouth became an accurate timer as it filled up with one milliliter of water every two seconds; a battery-driven Lego car moved precisely one centimeter in three seconds, so the dis-

tance it traveled also represented the elapsed time; a pendulum was carefully constructed with a swing time of precisely one second and was ingeniously electrically wired to blink a light bulb on each swing. Regardless of the material selected to construct their timers and clocks, the students were dealing with many of the same types of issues, such as accuracy and calibration.

BEYOND THE CLOCK PROJECT

A significant difference of the computer clocks became apparent in the area of extensibility. While many of the sand or water clocks were excellent timers, their use could not be extended beyond that. The computer clocks, however, were put to a variety of uses.

EXTENSIONS: A COMPLEX TIMER

In one instance, the students obtained a photoelectric eye, similar to the type used by stores to signal entrance and exit. Using an electronic interface box, they plugged it into the game port of a computer, and used a Logo command to measure the amount of light the eye was sensing. In another project, students were messing about with motion by using Lego blocks to build cars to go down an eight-foot ramp. Wanting the cars to be fast, the students experimented with design variables, such as the size and weight of the car, the diameter of the tires, whether more weight should be in the front of the car ("front wheel drive") or the rear ("rear wheel drive").

At some point, the students who were building Logo clocks realized that they could use their clocks in conjunction with the electric eye as timers for these Legomobile races. Now, different groups of students were working together, combining and expanding upon each others' projects. They placed the electric eye opposite a light bulb at the bottom of the ramp, and wrote Logo programs to measure the amount of time it took the cars to travel down the ramp. They placed a Legomobile at the top of the ramp, and let it go simultaneously as they began their timer programs. When the car reached the bottom of the ramp, it passed between the electric eye and its light source. The amount of light hitting the eye momentarily decreased, causing the timer program to stop. The last number printed was the time the car took to run down the ramp.

Realizing the inadequacy of whole seconds, the students expanded the computer clocks by improving their timer programs so that tenths of a second were printed out on their monitors (by changing the number following the WAIT command). This was probably the first time in their lives they had used decimals in a real and useful way. One boy, inspired by the Olympics, even tried to print out hundredths of a second. Some of the students were encouraged to calculate the speed of the cars in miles per hour. They did this

by knowing the length of the ramp and the amount of time it took the Lego-mobile to move that distance, and then converting to miles per hour. After many calculations (off the computer!) and spurred on by their own excitement and curiosity, they determined that their little cars were traveling at a rate of six to eight miles per hour.

The point of these explorations is that different groups of students had come together to solve problems in which they were interested. Some students had created a highly sophisticated timing device. Some had built the ramp, others the cars, two were experts on the electric eye, while others had written the functional Logo programs. Some had perfected the clocks to an accuracy of tenths of a second while others had calculated the speed of the cars in miles per hour. They had all joined in a rather informal way and had worked toward a common goal. To say that the computer was the central focus of the project is to miss the point, but it is clear that the extensibility of the computer to other objects was fundamental to the project's success.

EXTENSIONS: A TEMPERATURE SENSOR

Another area of computer extensibility was seen in some interesting work done with long-term Logo clocks used to measure aspects of the environment over periods of time. Some students, and the teacher as well, had long been concerned about the well-being of the class animals (hamsters, mice, snakes, turtles, and fish) during the cold winter nights, weekends, and vacations. Rumor had it that the schools saved money by shutting down their boilers at night, and it was feared that the animals would die or become ill as a result of the cold temperatures. We obtained a temperature sensor able to interface with a computer. The students wrote Logo programs that instructed their clocks to print out the reading of the temperature sensor every hour, and we set up the clock programs and left them running over the weekends. Students were relieved to find out that although the outdoor temperature measured in the teens Fahrenheit, the nighttime and weekend classroom temperature remained fairly warm.

Once again, the point is that while the computer was treated as another type of material in the classroom, it *did* possess the power to allow a link to be formed between the students' clocks and their very real concern for their animals.

ROLE OF THE CLOCK PROJECT IN THE COMPUTER CURRICULUM

It was not necessary for the students to be fluent with Logo to set up their clock projects. Indeed, they learned a good deal about computer program-

ming in the process of creating their clocks and timers. For example, students had been exposed several times to the idea of variables in a computer program. (A variable is a number that changes as the program proceeds.) However, only some of the students had assimilated this concept and used it in their programming. Others had not been so quick to grasp the idea of Logo variables.

When the students were confronted with the problem of making a timer, many of those using computers needed to find a way to represent seconds by a number that increased by one as the seconds ticked by. For the first time, they *needed* variables to solve a problem they were interested in. When they realized that they could solve it by using "one of those words with the two dots in front of it" (i.e., "seconds"), they understood the previously learned but not fully comprehended idea.

For most of the students, creating Logo timers was the first time they had used computers to make programs that made connections with the physical, tangible, noncomputer world. The insight that Logo could be used to solve real-world problems was further amplified when they used their Logo timers to determine the speed of their homemade cars and when they interfaced their clocks with the temperature sensor.

THE ROLE OF THE COMPUTER

WHY IT IS SO SPECIAL

The computer clocks, as compared with those made from other materials, were unique in several aspects. First, these clocks could be extremely *accurate*. As we have noted, some students calibrated them to tenths of a second. This high degree of accuracy is clearly unparalleled when compared with the other types of clocks the students made, and the students appreciated this accuracy when they wanted to time their Legomobile's speed precisely.

Second, it was quite simple to *adjust* the speed of the Logo clocks by changing the number following the WAIT command. Thus, it was relatively easy to match the handmade Logo clock to the clock on the wall or the wrist. By contrast, calibrating the sand clocks meant ripping them apart and changing the size of the hole through which the sand flowed. Only painstaking trial and error showed exactly how much larger or smaller the hole had to be. This was true for the other noncomputer clocks as well.

Further, the Logo clocks were more *adaptable* and could be easily *connected* to other ongoing projects in the classroom. We have described the timer/Legomobile connection and the clock/temperature-sensor experiment. The students took quite naturally to the integration of the Logo clocks with these other projects.

WHAT IS NOT SO SPECIAL

None of this should be taken to mean that the computer and Logo are the be-all and end-all of this type of exploration. Certainly the Logo clocks were accurate, adaptable, and easily adjustable—but the other clocks were wonderfully inventive, creative, and fun. They were also far more accurate than we would have predicted at the beginning of our study of time. While not as accurate as the Logo clocks, almost all of them certainly did the job, within a few seconds, of telling their inventors when the candle would go out.

It is important to note that none of the clock media (computers, sand, wood, etc.) stood out for the students as more desirable or valuable than the others. Some of the students were attracted to wood and so made their clocks out of wood. Similarly for water, or electrical devices, or pendulums. There was no competition for who could make the "best" clock (whatever that would mean) or even the most accurate one. There was simply a classroom filled with various types of clocks being constructed, some on the computer, some out of wood, some with water, and so on. All of the students were involved with their own, individual clocks, trying to perfect them to the best of their ability and interest.

A NEW WAY TO USE THE COMPUTER

We have described what we consider to be an example of truly educational computing—active, exploratory, student-directed learning involving the use of the computer. Through programming languages such as Logo, computers allow our students, within certain limits, to perform tasks that are difficult or even impossible to achieve with other materials. We emphasize that it *is* possible to create activities that connect many different students' interests to various curricular areas, and to connect "separate" disciplines to each other. Our goal was, and continues to be, to create learning situations in which connections are allowed to develop freely and to move in any direction, albeit many or even most of them unpredicted. A certain degree of openness and flexibility on the part of both teachers and students is obviously necessary to keep the inquiry interesting, stimulating, and exciting.

Some important guidelines, then, for the placement and use of computers in schools include the following:

1. Seek out open-ended projects that foster students' involvement with a variety of materials, treating computers as just one more material, alongside rulers, wire, paper, sand, and so forth.
2. Encourage activities in which students use computers to solve real problems.
3. Connect the work done on the computer with what goes on during the rest of the school day, and also with the students' interests outside of school.

4. Recognize the unique qualities of computers, taking advantage of their precision, adaptability, extensibility, and ability to mirror individual students' ideas and constructions of reality.
5. Take advantage of such new, low-cost technological advances as temperature and light sensors, which promote integration of the computer with aspects of the students' physical environment.

While the theme of this article has been the role of the computer in the educational process, let us clearly state that the ideas underlying our teaching strategies were formulated by educators and philosophers whose lives long predated the invention of the computer, and whose ideas can be applied to any learning situation and to any material. Our emphasis, as was that of Piaget, Dewey, Susan and Nathan Isaacs, and others, is clearly on the inquiry and the learner, not on the specific curriculum or facts to be learned. In this undertaking, all materials are created equal, although admittedly the computer did add unique and powerful aspects to the learning process.

Notes

1 "The Computer in Education in Critical Perspective," *Teachers College Record* 85, no. 4 (1984): 539-639.

2 Robert Taylor, ed., *The Computer in the School: Tutor, Tool, Tutee* (New York: Teachers College Press, 1980).

3 David Hawkins, "Messing About in Science," *Science and Children* 2, no. 5 (1965): 5-9.

4 Seymour Papert, *Mindstorms: Children, Computers and Powerful Ideas* (New York: Basic Books, 1980).

5 See Hubert Dyasi, African Primary Science Program, *Measuring Time: Part I—Making Many Simple Clocks* (New York: The Workshop Center, n.d., NAC 4/220, City College of New York).

Should Computers Know What You Can Do with Them?

DON NIX
IBM Thomas J. Watson Research Center

The use of computers in education can be an opportunity for children to surprise themselves and their teachers. The key is to empower the child with tools of self-expression.

The use of computers in education has the potential to contribute to the dignity of the child. However, as computers are typically used, this potential is neither met nor explored. In many cases, in fact, the educational use of the computer poses a threat to human dignity and uniqueness. A story by Woody Allen captures the nature of this threat. "My father was fired. He was technologically unemployed. My father worked for the same firm for twelve years. They fired him. They replaced him with a tiny gadget that does everything my father does, only it does it much better. The depressing thing is that my mother ran out and bought one."[1] This story blurs the distinction between machines and humans. Furthermore, it makes the machine attractive in a way that triumphs over the human. If one can imagine Franz Kafka surviving in the world of computers long enough to make a comment on this situation, he might add, as an extension to Woody Allen's story, "And my father went out and bought one too."

Dignity, then, for the purposes of my discussion, refers to the child's experience of himself or herself as intrinsically different from the way a computer functions, specifically by being able to actively consider his or her processes and feelings, and to be unpredictable in a creative way. (I am not arguing that this is what or all that dignity is, but merely being specific about my starting point.) Much of the use of computers in education currently poses a threat to this aspect of the dignity of children.

In order to focus clearly on this problem that computers pose, I will consider this question: Should computers know what you can do with them? When we discuss the problem in this context, the notion of creativity in the sense of unpredictability can be highlighted. In current computer applications in education, the computer to a considerable extent "knows" ahead of time what the student will learn in some content domain. That is, the computer as experienced by the child is structured so that one can predict to a high degree what the child will learn, specifically because of what the com-

puter is programmed to do. This predictability in fact is commonly used as a measure of the success of both the computer and the child. The danger here is one of mediocrity of knowledge. In addition to predictability in a given content domain, such as math, programming, or history, there is the likelihood of predictability in meta-cognitive ways. Based on the structure of the interaction between the computer and the child, the child learns a certain way, for example, of learning, or problem solving, or obedience. The experience contributes to the child's development of a conception of what learning is, and what his or her role is with regard to it, in terms of both cognition and modes of feeling. The danger here is a threat to dignity.

Presumably it is true that computers can have an effect on cognitive processing and feelings. This is a common and strategic assumption made both by people who are skeptical of computers[2] and those who see computers as at least potentially having a positive and liberating impact.[3] The truth of this assumption, however, has not been empirically supported by the normal experimental paradigms of psychology and educational psychology. Moreover, an assumption of this type, which involves complex issues of human functioning, is not a convenient one to test with such paradigms. Thus, based on evidence that is more experientially realistic than what can be provided by results in experimental psychology, I will assume that prolonged and/or critically important interaction with a computer can and does, in a somewhat Whorfian way, cause predictable changes in both cognition and affect in children.

In typical current applications of computers to education, then, computers know in an important sense what the child will learn. This predictability applies both to the content domain involved and to the way a child is influenced to think of and feel about learning and his or her role in it. The first poses the problem of mediocrity of knowledge, and the second constitutes a threat to the dignity of the learner.

EXAMPLES OF PREDICTABILITY

The most pervasive use of computers in education is the often derided, scorned, and ridiculed drill-and-practice mode, along with the related tutorial mode. These originated in traditional computer-assisted-instruction (CAI) days, over thirty years ago, and closely resemble programmed instruction technology, as well as the world of print workbooks and ditto sheets. In this mode, to the extent that the program is successful, the degree of predictability is high. The child learns prepackaged information (such as addition, decoding, history facts, science facts, and so on). The information is relatively simplistic, due to the exigencies of teaching in this mode of instruction. The meta-level of learning includes, among other messages, the notion that the concepts of right and wrong are central to thinking about learning, and

that these concepts can and should be unambiguously defined. The child also has the experience that he or she as a person is irrelevant to the process of learning. Learning is hunting for and remembering facts, rather than a process whereby the child explores, creates, and owns. The computer knows this.

The CAI modes are relatively clear examples of outcome predictability. Many objections to traditional and updated CAI have been expressed, different authors finding different features to excoriate. One particularly formidable objection is that of Papert.[4] A significant focus of this objection is the passive role the child is constrained to play in the drill-and-practice, tutorial CAI mode. Such a role is undignified. At best it ignores human potential. At worst it represses it.

An alternative plan for computers and education is exemplified by work by Papert and others, and embodied for example in the Logo language for children. In a Logo environment, the intention is for the child to program the computer, rather than being programmed by it. The child creates something with the computer. What is created can be a picture, or animation sequence, or, in principal, any kind of program that can be written with a general programming language. The computer is not pre-programmed to teach a certain set of facts about a certain content area. At this level, the computer cannot know what the child can or is doing with it.

At the processing level, however, the situation is different. Logo is a tendentious language. The language is designed so that, as an ideal, a child will learn a type of problem-solving technique. A significant goal of the Logo orientation is to have an impact on creative, self-expressive, problem-solving strategies. The goal is not for the child to learn unambiguous facts, but to learn processes that will not only be used to solve problems in subsequent Logo and Logo-like environments, but also to generalize to other domains in the life of the child. The problem-solving technique is computeristic. The domain of expression is defined basically in conceptual schemes related to computers, such as algorithmic thinking, procedural thinking, logical debugging, and modularization. Supporters of this type of role for the child at the computer define an important aspect of success in terms of the child's becoming an active problem solver of the type described. An ideal is for the child to master and become proficient at this type of creativity and problem solving in a wide range of domains beyond the specific computer setting that fosters it. It is at this processing level of predictability that the computer knows what the child will do.

Another area of computer use in education, and a recent development, comes from a noneducational field noted for its extravagant claims: artificial intelligence (AI)/intelligent computer-assisted instruction (ICAI). Proponents of artificial intelligence view the computer as "knowing" something in a more literal sense. In theory, the computer would, for example, under-

stand the content to be learned, the goals of the student, and the ongoing cognitive states of the student, with regard at least to his or her progress in learning what is to be taught. In a sense, then, the computer's knowing what the child can do with it is an explicit design goal. Again, success of the ICAI system can be measured in terms of a certain type of predictability. From the standpoint of technology, the ICAI approach and related "expert-systems" (or, more modestly, "paraprofessional-systems") approaches are radically different from the two referred to above, the drill-and-practice and the creative-programming approaches. From the dignity perspective, however, the differences are not that clear, although very few systems have actually been implemented for children, and even fewer have been explored in real-life settings. At the content level, the computer knows what the child will learn if the system is successful, similar to the situation with drill and practice, although the teaching methodology is quite different.

At the level of having an effect on the way a child might learn to think about problems, ICAI is basically an empirically unexplored area. It is easy to consider ways in which, for example, paraprofessional systems could be designed to teach either overtly or implicitly a manner of approaching some domain of problems, rather than or in addition to specific success in solving problems in that domain. However, the basis of such systems so far consists of explicit rules and algorithms for combining them. Given this deterministic type of system, no matter how complex, it is likely that the way in which a child learned to think and solve problems like the "expert" would be predictable. The computer would know.

THE COMPUTER KNOWS

In the above examples, arguments can be made that what the systems teach is worth teaching. This is not the issue. Any content materials can be taught in a variety of ways. The issue is whether these genres of activity between the computer and the child have side effects that are inimical to the child's exploring ideas and feelings that go beyond those comprehensible to computers, and do so in a manner of exploration that goes beyond the information-processing style of problem solving embodied in computers.

My assumption is that dignity is a desideratum. The above descriptions of several computer uses in education show that those uses do not exploit the computer's potential to foster dignity either fully or at all. Whether these uses are a boon, a bust, or Big Brother is not the issue. The point is that they are not conceptualized in terms of the type of creativity that is based on computeristic unpredictability. This does not mean they should be replaced. What it does mean is that there is considerable room in the world of exploring the uses of computers in education for a different type of system with different starting assumptions, and different effects. There is room for the type

of computer in education where the computer does not and cannot know what you can do with it.

THE COMPUTER DOES NOT KNOW

A paradigm we are exploring is intended to fit into the "computer does not know" category, in contradistinction to the ways computers are generally used in education. The term *Making a Scene* will be used to refer to the general environment and point of view we are implementing. This making-a-scene paradigm has not as yet been explored in detail. It is based on the use of a child-controlled multimedia computer system, including voice, video disc and VCR, audio cassette, animation, graphics, and a language named "Handy." The focus is on how the child can use the technology in an expressive way, rather than on the technology itself. The goal is to enable the child to be creative and self-expressive using the computer, but in areas that are not intrinsically related to computeristic concepts, and that cannot be expressed computeristically—ways that, in short, defy computeristic limitations.

This form differs from the traditional drill-and-practice systems, and the apparent aspirations of ICAI and paraprofessional systems, in that the stress is on self-expression. It differs from the programming-oriented systems such as Logo in that the areas of self-expression are more experientially familiar to humans. These areas include, for example, fun, humor, passion, satire, aesthetics, and interpersonal negotiations, in such topics as civil rights, rock video, political campaign advertisements, and television game shows. The making-a-scene computer uses the computer as a decentered participant, in contrast to most computers in the classroom. The computer does not know anything about a specific content area. The computer language does not intentionally embody a specific style of problem solving. The decentered role the computer plays prevents whatever hidden Whorfian determinants there are from being a focal point and a source of significant predictability. This is technology and clutter in the style of Max Headroom. It is an attempt to get closer to Papert's metaphor of the computer as pencil.

MAKING A SCENE

In order to explore ways that children could try out ideas using a computer, we created an experimental computer language. The language, Handy, will be briefly described here, and several case studies of its use will be sketched.

Handy enables a child (or other person as an author) to construct what can be called interactive scenes presented on a computer. The scenes can integrate video disc, video tape, audio tape, synthesized voice, digitized voice,

touch panel, animation, and graphics. The child can make the scene interactive according to whatever plan he or she has in mind. Examples of events include animating stories of interest, telling jokes, making a rock video or a play, and constructing an essay consisting of video and voice-over and text and touchable menus of questions that enable the reader-listener-viewer to browse this multimedia event.

Using Handy, the child creates the script and objects that are to be visible to whomever will subsequently interact with the scene or scenes. A script is a program written in the Handy language. The objects are individual elements that can be displayed on the computer screen. Objects can be pictures created by the child, or text, or windows onto video disc or video tape segments. Once the child has created all or part of a script and its associated objects, he or she can try them out in an interactive, iterative manner. Thus, once a scene or part of a scene has been created, it can be run, interrupted, revised, and continued. Scenes can be simple in the case of a small set of objects and a single script, or complex in the case of hundreds of objects and scripts that call other scripts or that run the objects themselves, instead of just showing them.

To take a relatively simple making-a-scene example, suppose a child wanted to comically annotate a few scenes from the movie *Ghostbusters*, and then show it to the class. The child might use a scenario as follows. A robot would come out onto the stage (the computer screen), and ask the viewer what he or she wanted to see—the school library, or the librarian. The robot would speak the question using the voice synthesizer, and also show two boxes for touching to indicate the choice. If the viewer touches the box indicating the library, the script would move the robot offstage, and then play from the video disc a scene from *Ghostbusters* showing the front of the New York Public Library. While this scene was playing on the computer monitor, the script could move onstage an object with text reading "This is the modest PS 233 School Library." When the scene ended, the text would disappear, and the robot would return and present the two choices again.

If the user touches the box indicating that he or she wants to see the librarian, the robot might be moved offstage, and the script would play the scene from *Ghostbusters* that shows a ghost drifting through the card catalog area. The label "Our Librarian" could be superimposed on the video. The video would then be scripted or programmed to freeze on a closeup on the ghost's face. A box would shrink down to frame just the face, blanking out the rest of the video image. The robot would come out. A text balloon would appear above the ghost's face with the words "I don't know why the children are afraid to check out books." Then once the user touches the screen, the box framing the ghost's face would close all the way up so that no video was visible, and the robot would ask the choice question again.

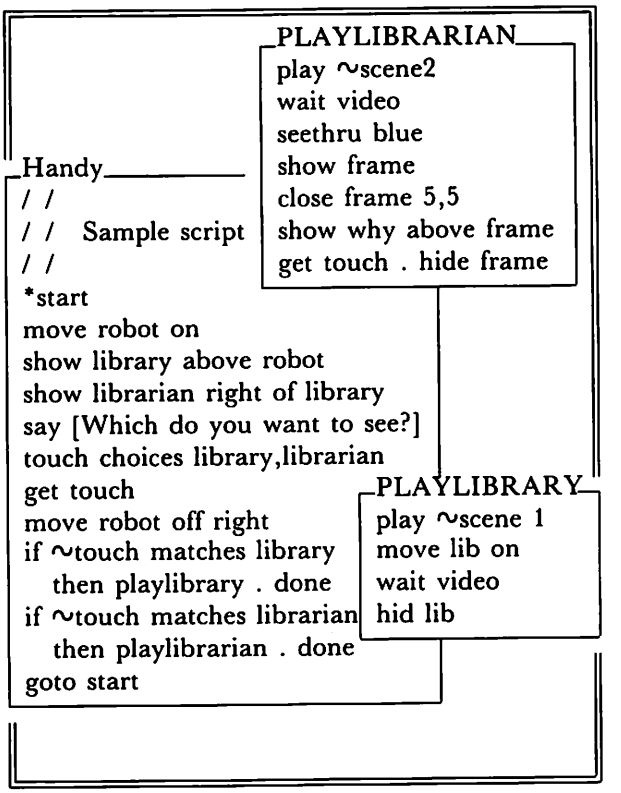


Figure 1. Sample scripts

In this example, which is based on a project done by fourth graders, the video scenes would be from the *Ghostbusters* movie video disc. The objects (robot, text balloons, frames, and so on) would be created by the child, and superimposed on the video material. An example of a script written to perform the above is shown in Figure 1. What each object shows is as follows (uppercase words are the names the child has given to the objects created and appear in Figure 1 in lower case): ROBOT is the robot; LIBRARY contains the text "I want to see the library"; LIBRARIAN contains the text "I want to see the librarian"; LIB contains the text "This is the modest PS 233 School Library"; FRAME is a box for outlining the ghost's face; and WHY contains the text "I don't know why the children are afraid to check out books." Much of the script is self-explanatory. The statement "∿touch," however, needs explanation. This statement is a variable that is automatically replaced by the word "library" or "librarian" once the person interacting with the scene

has made a choice and touched one of the objects, LIBRARY or LIBRARIAN. If the LIBRARY object is touched, for example, then the script determines that "∿touch" matches the word "library" and so the script runs the subscript labeled "playlibrary" via the statement "then playlibrary." The "playlibrarian" script is run if the LIBRARIAN object is touched.

The variables "∿scene1" and "∿scene2" contain the video disc frame numbers for the two scenes the script will play. These are set by the child by viewing the video, and then telling Handy what sections of the video are to go by those names. Any names could be chosen.

The example in Figure 1 is relatively simple. Handy itself has hundreds of additional commands and facilities, and can be used for quite sophisticated programming. The application of Handy in Figure 1 uses only a small subset of the system. It is presented here to indicate the type of language Handy is. However, even in this example, the child has to study the movie; analyze it; reconceptualize it; create a mode of presentation, including a mode of interaction, for the intended audience; plan and create the objects; and write, try out, and revise the script. The creativity involved in doing this is not circumscribed by computeristic concepts. It involves a wider range of issues, many of which are "real world" issues that are unrelated to computers, such as humor, satire, and a challenge to authority. On the other hand, the child does deal with computer concepts quite directly in order to create his or her production. The script in Figure 1 *is* a computer program. It is not English. It is written in an artificial language that has to be learned, using where possible what one knows about English as a guide. Once the script is written, or some of it is written, it has to be debugged.

The design goal of Handy was to create a language that would facilitate a type of noncomputeristic self-expression. The intention was to make the language itself as decentered as possible. As children use Handy, their experience will provide information needed to revise it. We have begun a series of exploratory studies in which children create events with Handy. Although there are many questions that can be experimentally studied using this environment, at this early stage the focus is on providing children with the facility, and seeing what they do and where they have problems. Several examples will be briefly outlined here.[5]

One class of Handy use is that of creating interactive video essays. In general, a child creates an "essay" that, instead of being written on paper, consists of video segments chosen by the child to make his or her point, combined with superimposed annotation or extended text, and voice-over from the child. The person who "reads" the essay actually reads and listens to and watches the essay, and can make choices about what he or she wants to see next, or see again. The *Ghostbusters* example above is a simple example of this type of essay. A more elaborate example is a project in which tenth graders created mock political advertisements for a presidential candidate,

using video-tape footage from *The Video Encyclopedia of the 20th Century*[6] consisting of seventy-five hours of newsreel footage containing news clips from 1893 to 1985. In this project, students picked presidential candidates (Ford and Reagan) and used film clips from their campaigns as raw material for the video component of the essay. In addition to the footage from the actual film clips, the students shot additional footage themselves, and then edited it all together.

In creating the essays, a number of decisions had to be made by the students. Examples include: demographics of intended audience; relevance of the ad for the overall campaign; emphasis (record of achievement, personal qualities, platform, positive versus negative campaigning, and others); emotional tone; thematic integration; use of media (camera angles, based on what is available, sound, voice-over, annotation, graphics, and others). The students implement their decisions by creating objects consisting of text and pictures and other types of information, and writing a script to put it all together. For example, at one point in the Ford essay, the script was written to play a video segment showing a Fourth of July celebration with fireworks displayed on a dark sky. The script also showed a computer-generated American flag superimposed on the video, and at the same time played an excerpt from Beethoven's First Symphony. The computer did not know what the students had done with it.

A separate but related essay project used a video disc containing civil rights footage from the 1960s. The video was excerpted from several of the *Video Encyclopedia* video tapes. Eighth graders used shots of speakers (for example, Martin Luther King, Jr., George Wallace, and Julian Bond), as well as police and crowd scenes (for example, federal troops sent to Little Rock, Arkansas), to express their ideas about civil rights issues. For example, for video footage that has no sound with it, a student can record narration on a computer-controlled audio tape, and then write a script to play the video with their narration at the same time. As another possibility, the script could be written to allow the person who is listening to the essay to record his or her own voice over the video.

In these projects, the students actively used a computer and a specialized environment to produce essays that could not feasibly be produced without computer support. The computers were an integral part of one aspect of the expression, but not a focal point in the overall experience (similar to the pencil analogy). In this interactive video type of essay, the computer does not know what the child will create, or think, or learn. The most significant aspect of what the child does is not related to concepts about computers. The civil rights essay was a moving multimedia interactive presentation that was a revelation to youngsters for whom the 1960s did not previously exist. The Ford campaign was for some of its audience a persuasive advertisement, and for others a source of satirical hilarity.

Another form of self-expression is a project in which a group of fifth-grade children created a "disc jockey" program, where a viewer could request a

song. They created an object representing a disc jockey, including mustache, Mohawk hair cut, and sweater with "WPLJ," their favorite radio station's call letters. They also created a menu that had four songs, "Private Dancer," "West End Girls," "Working on the Highway," and "I Call Her Darlin', but She Calls Me Collect." They wrote a Handy script for a scenario in which the disc jockey bounced onstage, and the menu appeared, and the disc jockey said, using the voice synthesizer, "Pick a song." The children had created their own video tape. On the tape they were singing and dancing to each of the songs. They had made costumes, picked the songs, done the choreography, and handled the production details. Then they viewed the tape through Handy, in order to assign the segments they wanted to show for the different song choices their viewers would make. Once the user made a song choice, the disc jockey would bounce offstage, the script would locate the beginning of the song on the video tape, and an object would open up on the screen, showing the singing and dancing. The script was written so that, at any point in time, the user could interrupt the song and dance, by pressing a certain key, and then return to the main menu.

This project involved a considerable amount of group interaction and planning, from picking the songs to scheduling time when all four could get together after school to shoot the video. Again, the point is that the focus is not on math drill or computer programming, but on using a computer environment to create (in this case) a rock video, where the most important elements involved in the creation were social interaction, aesthetics, music, humor, popular culture, and production.

Two more examples will be mentioned, which are more ambitious in scope. The first is a soap-opera project done with eighth graders from the Dalton School, and appropriately called "Dalton Crest." The students wrote a screenplay for the story that involved a complex set of personal relationships among six people, and in which a murder occurred. The screenplay was written so that each character had a plausible motive for the murder, but so that one character was implicated in subtle ways. We videotaped the soap opera, and each of the creators played a role in it. A viewer of the soap opera encountered the following situation when he or she interacted with it on the computer. A short initial video segment was shown, with computer-generated text, to introduce the characters and establish that a murder had occurred. Then it was up to the viewer to solve the mystery by requesting more information. The additional information was presented in terms of video segments. Background and other information was superimposed on the video. At any point the viewer could offer a solution. The Handy script kept a record of how the viewer traversed through Dalton Crest.

This project involved a wide range of cognitive, affective, and interpersonal activities. The computer played a part in enabling the students to think about a range of issues, and then to be able to creatively explore those issues in a highly motivational, personally meaningful, and publicly entertaining way.

The final example was done by a group of children, grades 3 through 5. The project began simply as an effort to teach a version of Handy to the children, in order to determine where there were problems in Handy that had to be corrected before the language could be used on a wider scale. This project evolved into an elaborate satire of a popular television program, "Family Feud." On this show, two families vie against each other by trying to guess how people on a previously conducted survey answered certain questions. The five or so most frequent answers are on a board, but covered up. If a family member guesses one of the answers on the board, the answer is uncovered. If the guess made is not one of the answers on the board, an X appears and a buzzer is sounded. Key elements in the television program are the fawning attitude of the moderator and the way in which members of the two families overreact to the moderator, the questions, the answers, and everything else. The name of the project the children did was "Family Fools."

In this case, the computer was used to make an "answer board" like the one on the show. The children created the objects for the answers to questions, and objects to cover them up with decorative patterns. A script was written to check for an answer to be typed in, and, if there was a match (incorrect spellings and other types of approximate answers were accepted as correct), to uncover the answer, and, if not, to show one, two, or three Xs, and sound a buzzer. The production of "Family Fools" included the children, the sychophantic moderator, and the computer. The children divided themselves into two different families. I was given the role of moderator. Two video cameras were used, one for the moderator and families, and one focused on the computer. The two video sources could be mixed, or one could be inserted into a corner of the other. The show was taped before a live and motley audience consisting of children waiting for their turn on camera, researchers, and passers-by who wandered in because of the noise.

The "Family Fools" project, although begun as a computer activity and based in large part on a computer program ("script"), was considerably different in content and feeling from a computeristic activity. The computer was a catalyst and tool for a set of creative efforts that were experientially familiar to the children, that were expressed creatively, and that reached into diverse content areas.

The reason for sketching these representative projects in some detail is to give as tangible as possible an indication of how this kind of use of computers in education differs from current uses, and from most proposed uses, and as suggestions for further exploration. On consideration of these descriptions, or observations of or participation in these and other related projects, it is obvious that something quite different is going on, when compared with what one would normally observe in schools or research laboratories. The computer is a decentered enabling technology for experiences whose significance transcends computeristic ideas.

CONCLUSION

There are two burning and unresolved issues that are particularly relevant to the type of computer environment described here. I want to place the making-a-scene environment in perspective with regard to these issues, without direct concern for the resolution of the issues themselves.

First, do or can computers have any effect at all, other than simply teaching a mediocre level of content knowledge in some specific domain? Writers who have considered the good and the evil of computers in society, and for children more specifically, including Papert, Weizenbaum, Turkle, Dreyfus and Dreyfus, and most contributers in Sloan, as well as Woody Allen and comic denizens of late night television, assume that computers do have such effects as a result of direct interaction with them over a period of time.[7] Little if any empirical evidence is given of the type usually considered necessary to establish a cause-and-effect relationship. This has not been empirically tested in a fair manner to a satisfactory extent, and perhaps cannot be, given the limitations of the test paradigms used, which are usually based on experimental psychology. The relationship is assumed, and the arguments start there, in terms of whether the results are good or bad.

This type of question as applied to the making-a-scene genre of computer activities is shifted somewhat. The focus is not directly on whether the computer makes a difference. It is on whether the nexus within which the computer is decentered, and in which activities involving aesthetics, social interaction, humor, and so on, makes a difference. In this environment, it is easier to assume that a difference is made, because the young people are more totally involved, and they more directly own what they are doing, in terms of both the cognitive and the affective elements. It has in general been shown that the deeper the processing of information, the wider the range of types of processing, and the greater the motivation and sense of ownership, the greater the impact. The working assumption, then, is that computers in education, à la making-a-scene, can have a significant impact.

The second issue to be considered is: Are computers intrinsically limited and different, compared with humans, in terms of mentation and affect? From a theoretical point of view, it is difficult to see how this question can be resolved in the near future, despite the increasing amount of discussion and the strength of conviction many of these discussions demonstrate. If it seems that humans have an element of freedom, then the intrinsic differences are clear. If, on the other hand, it makes more sense to believe that humans are deterministic creatures, then the differences between computers and humans are less intrinsic and more a matter of complexity. However, despite the theoretical difficulty in resolving the issue, from a more immediate and more practical viewpoint it is clear that computers and children are cognitively and emotionally different. Computers cannot create soap operas that elaborate complex interpersonal relationships and act them out with convic-

tion, and cannot experience strong feelings when confronted graphically with the realities of the racism of the 1960s. Computers represent a "cognition" that if evidenced in a human would be classified as clinically retarded.

My concern is not that computers may ultimately be as smart as or no different from the brain of a child. My concern rather is that computers as they now exist in education are either not conducive to, or limit, dignity. The making-a-scene genre of computers and children is a means of dealing with these concerns. In a meaningful way, the computer does not know and cannot know what the child will learn. The type of computer-child interaction using the environment described is an additional way to study the impact of computers on education. With a computer that does not know what the child learns we avoid some of the concerns about inimical effects of computers on dignity. More positively, this is a way of considering a wider range of potential enhancements of education by computers. The use of the computer is creative; the creativity is not predictable; and the child in his or her interaction cannot be replaced with a tiny gadget.

Notes

1 Woody Allen, *Album Number Three*, New York: Capital Recording Company, 1965.

2 See for example, Joseph Weizenbaum, *Computer Power and Human Reason* (San Francisco: W. H. Freeman, 1984).

3 Seymour Papert, *Mindstorms: Children, Computers and Powerful Ideas* (New York: Basic books, 1980), supports this point of view.

4 Ibid.

5 I am grateful to children and staff at the Dalton School, New York City, and to children from the Tarrytown, New York, School District, for these examples, and the experiences we had working on the projects.

6 Available from CEL Educational Resources, 515 Madison Avenue, New York, NY 10022.

7 Papert, *Mindstorms*; Weizenbaum, *Computer Power and Human Reason*; S. Turkel, *The Second Self* (New York: Simon & Schuster, 1984); H. L. Dreyfus and S. E. Dreyfus, *Mind over Machine* (New York: Free Press, 1986); and Douglas Sloan, ed., *The Computer in Education: A Critical Perspective* (New York: Teachers College Press, 1985).

Will There Be Teachers in the Classroom of the Future? . . . But We Don't Think about That

SETH CHAIKLIN
Teachers College, Columbia University

MATTHEW W. LEWIS
Carnegie Mellon University

At first, new educational technologies engender reactions pro and con. As the new possibilities mature, however, different issues become more clearly defined, the classic problems of education—the responsibilities of teaching, the selection of content, the justification of competing goals, the mundane mechanics of implementation, and the inspiration of unstinting effort.

Scientists Teaching Computer to Drive a Car

Associated Press—August 27, 1986

PITTSBURGH—Carnegie-Mellon Univeristy scientists are working on a computer nicknamed "The Warp" that they think will one day drive a car.

The computer, about the size of a refrigerator, has driven a small cart at less than 1 mph, according to Dr. H. T. Kung, a professor. It uses a television camera to see the road and common traffic obstacles. . . .

Later generations of the computer may be used to help handicapped people who need some assistance in driving, according to Richard Cyert, Carnegie-Mellon's president.

We would like to thank Rick Werthiemer, mathematics instructor in the Pittsburgh City Schools, for his time, insights, and helpful comments on this chapter. We would also like to thank Marilyn Jacobs for her critique of an earlier draft. The tutoring projects described here are products of John Anderson and the ACT tutoring project teams. Matthew Lewis is the leader of the Teacher's Apprentice Project, a project with a goal of producing a tutoring architecture for several topics in high school mathematics. As a member of Anderson's laboratory he has been associated with the Geometry and LISP Tutor projects for several years. Seth Chaiklin did not actually participate in the tutoring projects. This chapter is based on a paper that appeared in Jon Jacky and Doug Schuler, ed., Proceedings of the Directions and Implications of Advanced Computing Symposium, *Seattle, Washington, July 1987.*

"The Department of Defense would like to think it would be good to run a tank with it, but we don't think about that," Kung said.

THINGS WE DON'T THINK ABOUT

The preceding news item illustrates dramatically how artificial-intelligence (AI) applications can raise questions well beyond the scientific goals that originally motivated the application. AI research that presumes to provide some societal application necessarily enters into an arena of questions concerning social choices and effects, and it is the responsibility of AI researchers to acknowledge and address the host of societal questions that arise when we work in socially relevant domains. Our contention is that it is irresponsible to deny or ignore these social questions. Such denial is often justified in terms of a separation of scientific from societal questions. We cannot have it both ways: If we presume to argue for the social utility of AI applications, we must accept the responsibility to address their social implications.

This article develops this general theme by examining the application of AI in the form of intelligent computer-assisted instruction (ICAI). In the educational realm, AI has crossed that vague boundary from theoretical work to practical work. Having done so, AI has also entered into a new realm of responsibility whether it is acknowledged or not.

Our concern here is to make an initial survey of the topics and issues that might profitably be included in a more systematic analysis of impacts of ICAI on education, as well as some specific suggestions based on the current state of ICAI. We do not pretend to have a complete analysis. This article is a first step in practicing what we are preaching.

To make our analysis, we first examine typical claims that have accompanied the application of AI to education and identify incumbent responsibilities that emerge as a result of this shift from laboratory work to an arena with social concerns. Then we consider two cases in which ICAI has been used for educational purposes. These practical applications enmesh us in issues that go well beyond the original scientific questions: the social costs of developing ICAI (allocation of resources), the pedagogical limits of ICAI relative to the educational goals of society, and system designs that are sensitive to the practical constraints of school classrooms. Finally, we make some observations about issues that were not raised directly by the analysis.

Our conclusion is that the AI community must encourage and support forms of communication that have not traditionally been a part of our science.[1] As in other scientific communities, the usual standard is to report what is new relative to the existing knowledge base. We see the need to develop an expectation that researchers learn about the social context in which they are applying AI and develop some sensitivity in analyzing the implications of their application relative to social goals and expectations. In the case

of ICAI this means investigating the practical dimensions of applications of ICAI in educational contexts, much in the same way that engineers must consider the environmental impacts, economic constraints, and safety factors in their design of bridges, cars, and toasters. Should our designs of human environments be done with any less care?

We do not think that anyone will find this suggestion of critical evaluation novel nor problematic. However, we want to call attention to the fact that there is not now an expectation that critical evaluations should be done by the same community that generates ICAI applications. At the same time, we do not believe that it would be desirable for AI researchers to be the only voices to contribute to the analysis of ICAI in the classroom, nor must all researchers who are making experimental applications of AI to education discuss the social implications. We do think, however, that claims about the social implications of ICAI should be subject to the same rigorous evaluation and criticism that the researcher would apply to the scientific parts of the work. This includes careful consideration of extrapolations from existing data, populations, and settings.

Who is in a better position than the researchers themselves to interpret the potential of ICAI as well as its limits? Researchers have a responsibility to provide a realistic assessment of the accomplishments of their research when it purports to address socially significant institutions. We should not expect that, following a simple description of the content of ICAI systems, educators will understand how to coordinate such systems with their existing forms of education and their curricular requirements (which are often legislatively mandated and enforced by annual testing), and be able to identify gaps in the ICAI systems relative to educational goals. If AI researchers are not knowledgeable about these issues and skilled in their analysis (i.e., unless we start "thinking about that"), the ability of AI researchers to make sensible contributions will be hampered.

Such awareness and abilities are not developed in a vacuum. They are unlikely to happen if they are not expected, supported, encouraged, and rewarded within the research community. We would like to see the time come when social analysis of ICAI discussion is an expected part of the work. It should not be considered a waste of time or a "necessary evil." It is important for us to demystify, interpret, and explain ICAI systems to the communities into which they will enter and to the public. Part of the common understanding of systems must be an appreciation of their strengths and limits.

CURRENT VIEWS ABOUT THE EDUCATIONAL POTENTIAL OF AI

We start our analysis by considering the kinds of statements that have been made about the potential of ICAI. Our intent is to briefly sketch some cur-

rent views and claims about ICAI's potential because they set a tone for discussion and expectation within the scientific community. In turn, these issues percolate into the general public understanding.

What have the pundits and Cassandras predicted for the cross-fertilization of AI and education? The general view is that major changes are at hand. In 1983, at the request of the White House Office of Science and Technology Policy and the National Science Foundation, the National Academies of Sciences and Engineering and the Institute of Medicine organized a panel of distinguished AI researchers and cognitive scientists to provide a briefing on cognitive science and AI. The area of AI in education was described as showing "great promise for making fundamental advances and an area that is capable of making solid contributions to the nation's education problems."[2] Psychological knowledge about how skills are acquired, combined with AI techniques and affordable, powerful hardware, had made it possible to design and develop effective "intelligent" educational software.

The scientific community, despite its high expectations for ICAI, has taken a relatively moderate stance about what has been accomplished. This is particularly notable and encouraging given the notorious reputation of AI claims relative to AI accomplishments. Consider some of the more optimistic claims that have been made for ICAI. A subheading in a *Byte* article about an intelligent LISP tutor reads: "It approaches the effectiveness of a human tutor."[3] The implication that teaching machines will teach as successfully as good human teachers reflects claims of other researchers in the ICAI area that intelligent machine tutors will replace human instructors for a large part of the curriculum. Computer-based tutors that teach problem-solving skills to students are very different from a tutor that explicitly teaches conceptual knowledge for a substantial part of a curriculum. To the best of our knowledge none of the latter currently exist. We still have much to learn about how to tutor conceptual knowledge in complex domains.

Similarly, Brown refers to the "revolutionary impact" of powerful new hardware in computer-based tools for learning.[4] In his view, learners will be free to develop domain-independent problem-solving skills and "meta-cognitive" skills. The nature of these skills and how they develop is currently very poorly understood. It is safe to say that teaching these skills is a goal that human teachers have yet to get any successful handle on, and cognitive science has yet to come to grips with in any substantive way. By successfully tutoring domain-specific problem-solving skill, we may be focusing on those aspects of education that are relatively easy to teach. We must be wary of the extrapolation of successes in one area to future successes in other areas.

Other researchers give a more conservative, but clearly optimistic view of the positive impact of ICAI on future education.[5] With some caveats about the social impacts of powerful computers[6] and their effects on classrooms,[7] the overall view is very bright: The new wave of high-powered hardware and

sophisticated software will revolutionize education. Predictions for change generally include changes in the role of the teacher and classroom structure, the way we structure curriculum, the way we set educational goals, and the way we educate teachers.

ICAI HAS BEEN IN THE CLASSROOM

We need not wonder about whether ICAI will have effects on the classroom and education as outlined above. In several settings ICAI has been in the classroom, employed in teaching substantial portions of the curriculum. In general, these ICAI systems run on work stations and are organized by a simulation of human problem-solving behavior. These systems allow varying amounts of exploration during problem solving and can provide hints and analysis of problem-solving attempts and errors in real time. The LISP Tutor[8] has been in regular use at Carnegie-Mellon University since 1984 and available commercially since 1985. It teaches a semester-long course in LISP programming and has been used in the instruction of both technical and non-technical undergraduates, as well as commercial and government programmers. The Geometry Tutor[9] has been integrated into a high school geometry classroom for a total of one school year teaching proof-solving skill. An intelligent tutor for algebra equation solving was introduced into a high school classroom in the spring of 1987.[10] For this small population of students and teachers the future is now.

The learning results reported by the researchers are promising: time savings in the range of 40 percent, and significantly increased problem-solving ability for both the LISP and Geometry tutors. The data are not yet available from MacArthur's algebra tutoring study. On the basis of such results, it would seem that all is right with implementing ICAI. However, we believe the situation is not that simple.

IMPLICATIONS BASED ON THIS EXPERIENCE: WHAT WE CAN SAY

In light of these applications of ICAI and as part of our claim that extratheoretical concerns in research are legitimate and necessary, we want to highlight issues that are not typically considered in the predicted areas of change noted above. Our discussion shall consider both the Geometry Tutor and the LISP Tutor.

Teacher Role and Classroom Structure

Both the role of teacher and the classroom structure changed significantly when ICAI went into the classroom. Teachers have given some of their tasks

to the machine (such as evaluating solution of practice problems); they have had to reorganize what they teach and the amounts of time they spend on different activities; and they have taken on new tasks, such as servicing the hardware and maintaining the software.

Dramatic changes in classroom structure result from the fact that the tutors are good at helping students practice procedural skills and at helping students overcome difficulties in executing these procedural skills. Because the tutors do not explicitly tutor the conceptual ideas that organized these skills, this has also affected the role of the teachers in classrooms where ICAI has been introduced.

In the geometry ICAI classroom, the teacher alternated a week in which new concepts were introduced in a traditional lecture format with a week of using ICAI to let students refine their skill by applying the new concepts to solve proofs. Proofs were not normally assigned to be solved during class in the pre-ICAI days because students got stuck early in their solution attempts. Teachers often were not able to provide the in-depth assistance needed to help "unstick" individual students and at the same time maintain an orderly classroom. The Geometry Tutor, however, helped students when they got stuck, so classroom behavior problems were manageable and the teacher could assign proofs in a class with a one-to-one ratio of students to computers. Because students could work on proofs in class in a productive way, the teacher reported that he had more time to deal with the individual needs of students. He could take time in the classroom to both attend to the problems of below-average students and challenge his more advanced students.

In typical secondary math and science classrooms, teachers are expected to convey subject-matter concepts, help students tune their problem-solving performance, and guide the application of the acquired skill to novel problems. However, to realize these goals, teachers usually have to create a supportive atmosphere, motivate students, act as implicit role models, and deal with the affective needs of and variations among students, among other things. Some of these motivational concerns may be reduced in a university context. The classroom structure in a university environment that uses the LISP Tutor is comparable to self-paced courses. The tutors do not take over the instruction of conceptual knowledge. This is obtained from a specially prepared textbook,[11] and the students attend introductory lectures on how to use the system. Students are instructed to first read a chapter from the textbook. They then go to the terminal room and use the dedicated terminals to be tutored on the solution to the programming exercises at the end of the chapter. After certain chapters they use the system to take quizzes. Only the midterm and final examinations are taken with pencil and paper. After the initial "how-to" lectures, contact with an instructor takes place in the form of student visits during office hours, in which the instructor can clarify con-

cepts and their application. The teacher is now a learning consultant (answering conceptual questions and helping to debug serious misunderstandings).

Beyond these changes in pedagogical tasks, teachers using both the Geometry and LISP tutors have to deal with software and hardware maintenance. Even in the classrooms that contain ICAI machines, our experience has been that machines break or get unplugged, printers are off-line, connections are erratic. Maintenance routines sometimes require a fairly complex knowledge of the operation of a system. A teacher must acquire some of these skills to keep his or her class running.

Curriculum Structure

We have seen how the role of the teacher and the structure of the classroom have changed, but what about the changes that were made in *what* was taught, and the order and pacing of the subject-matter presentation? One of the major promises of both CAI and ICAI has been that "students can work at their own pace." For the Geometry Tutor this is not the case. By definition, if the geometry curriculum is being taught at the blackboard every other week, the class must progress at the same rate. Students are not allowed to work at their own pace on the scale of the overall curriculum. They are free to solve the problems within a lesson as quickly or as slowly as they wish, but the class will go on to new information the next week. Granted, the pace of the class was accelerated, but the problems and order of presentation of the curriculum were not individualized.

The case for individualization and self-pacing with the LISP Tutor is different. Like other self-paced courses in university settings, it appears to be successful in adapting to differential learning rates. There is no problem with some of the constraints found in high schools: limited access to machines due to classrooms being locked at 3:00 P.M. and the insurance liabilities of having students in classrooms without instructors.

The point is that individualized pacing is a promise of ICAI that has yet to be applied in a high school setting, and may have some fundamental limitations due to the structure of traditional classrooms and access to machines. We must again check our extrapolation of one experience to another without analysis of the constraints in each situation.

Educational Goals

Fact: Geometry and LISP programming students are learning more by using the ICAI systems than they learn in "normal" classes. Generally, students improve their grades by one level: C students in normal classrooms are performing like B students with the help of computer tutoring. This is not sur-

prising given the results presented by Bloom, which show that students who receive one-on-one human tutoring usually perform two standard deviations higher on comparable tests than students who are taught the same material in a typical classroom setting.[12] If the combination of the teacher and the ICAI system is providing a good deal of individual tutoring, then the improvement of a letter grade is not unexpected. Given that students can acquire certain skills more quickly and effectively, what do we want to teach? We will address this question below.

Teacher Training

After participating in the "training" for use of a major hardware and software traditional ICAI package in a public school district, we are able to see more clearly what is involved in the application of ICAI. Proper training is of key importance. In the case of the Geometry Tutor, the teacher who did the majority of the teaching was brought in for a summer of experience working with the hardware and software—how to deal with the unexpected hardware and software problems, how to change teaching style, how to run multiple students through a single machine, how to lay out a classroom, and so forth. These are critical skills that we should not expect teachers to invent or induce. These issues should be well thought out, specifically defined to be directly applicable to classrooms, and they should be practiced. Changing teaching style is by no means a trivial endeavor. Teachers need time and instruction to work with different teaching styles, especially adaptive-individual or small-group tutoring. We should have a feeling for the range of metrics that teachers use (and are trained to use) to evaluate student progress so that these metrics and possible training simulations can be designed into the system, instead of being tacked on at the last minute.

These practical aspects might incline ICAI designers to be sensitive to such issues in their designs. For example, one might design teacher tutorial components into the system and include classroom teachers on the design team to get their input as to what the system's history of each student should include.

RELATED CONSIDERATIONS OUTSIDE THE LIMITED CURRENT EXPERIENCE

There are issues relevant to the four topic areas discussed above that were not encountered substantially in our direct experience with ICAI.

Motivation

One of the biggest hurdles in many teaching situations is motivation. Some research has identified principles that appear to increase the motivational

properties of ICAI.[13] Examples include making systems highly interactive and having varying levels of challenge and achievement. However, it comes as no surprise to find that current and projected ICAI is not able to simulate the motivational and affective sensitivities of a good teacher.

Coordination with Curricula

One of the problems encountered with all teaching aids is coordination of the aid's curriculum with the curriculum of the individual teacher or the district. Should we also be thinking about how to design the curriculum so that it is flexible enough to be tailored by districts and by individual classroom teachers, without needing programming expertise? How might the curriculum and tutors adapt themselves to the curriculum and language of the teacher and not vice versa?

What Do We Want to Teach?

Some issues will not impinge directly on the design of specific instances of ICAI, but they do help shape the general goals for designing such systems. For example, if we are able to teach all the high school mathematics curriculum in three years, then what other topics do we want to teach? More applications? Similarly, we have to be clear about how our systems interact with current educational goals. For example, will teachers be able to say "I don't have to teach anymore!"? Will districts start to accept the standards of what can be taught easily with ICAI (currently problem-solving skill) instead of asking what they want students to learn and asking what the tutors *do not* teach? These are questions that we believe system designers should be addressing when they talk about their systems and about which they should be generally knowledgable.

THINGS TO THINK ABOUT

To reiterate the main point of this article: ICAI systems are being introduced into school classrooms and will continue to be introduced in the future. Their introduction confronts us with the common issues that face any educational innovation: the role of the teacher, the subject matter to be taught, the educational goals for the students, teacher training that might be needed to work in this new setting, and motivation of the students. ICAI developers have focused their efforts on the clear communication of subject-matter knowledge. However, we must not let the sophistication of the machines and their cognitive models dazzle us into thinking that we can avoid the complex problems that accompany educational innovation.

We have just begun to think about the implications of actual cases of ICAI in the classroom. We hope that this conversation will continue among all

those who have a stake in the matter. Conscious efforts will have to be made to support such dialogue however, because there is little support and expectation for it to happen at present.

Notes

1 However, Computer Professionals for Social Responsibility and their 1987 symposium on directions and implications for advanced computing are positive steps in this direction.

2 W. K. Estes and A. Newwell, "Report of the Research Briefing Panel on Cognitive Science and Artificial Intelligence," in *Research Briefings 1983* (Washington, D.C.: National Academy Press, 1983), p. 32.

3 J. R. Anderson and B. J. Reiser, "The LISP Tutor," *Byte*, April 1985, pp. 159–75.

4 J. S. Brown, "Process versus Product: A Perspective on Tools for Communal and Informal Electronic Learning," *Journal of Educational Computing Research* 1, no. 2 (1985): 179–201.

5 O. Park, R. S. Perez, and R. J. Seidel, "ICAI: Old Wine in New Bottles—or Is It Vinegar?" in *Artificial Intelligence and Education: Issues and Applications*, ed. G. P. Kearsely (Reading, Mass.: Addison-Wesley, 1987), pp. 2–45; D. Sleeman and J. S. Brown, "Introduction, Intelligent Tutoring Systems," in *Intelligent Tutoring Systems*, ed. D. Sleeman and J. S. Brown (New York: Academic Press, 1983), pp. 1–11; R. P. Taylor, "Introduction," in *The Computer in the School: Tutor, Tool, Tutee*, ed. R. P. Taylor (New York: Teachers College Press, 1980), pp. 1–11; and A. C. Wilkinson, "Issues at the Interface of Theory and Practice," in *Classroom Computers and Cognitive Science*, ed. A. C. Wilkinson (New York: Academic Press, 1983), pp. 3–13.

6 Brown, "Process versus Product."

7 M. Amarel, "The Classroom: An Instructional Setting for Teachers, Students, and the Computer," in Wilkinson, *Classroom Computers and Cognitive Science*, pp. 15–29.

8 Anderson and Reiser, "The LISP Tutor."

9 J. R. Anderson, C. F. Boyle, and G. Yost, "The Geometry Tutor," in *Proceedings of the International Joint Conference on Artificial Intelligence* (Los Altos, Calif.: Morgan Kaufmann, 1985), pp. 1–7.

10 D. McArthur, C. Stasz, and J. Hotta, "Learning Problem-Solving Skills in Algebra," *Journal of Educational Technology Systems* 15 (1987): 303–24.

11 J. R. Anderson, A. C. Corbett, and B. J. Reiser, *Essential LISP* (Reading, Mass.: Addison-Wesley, 1987).

12 B. S. Bloom, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher* 13, no. 6 (1984): 4–16.

13 T. W. Malone, "Toward a Theory of Intrinsically Motivating Instruction," *Cognitive Science* 5 (1981): 333–69.

Contributors

JOHN P. BLACK is associate professor of computing and educational psychology at Teachers College, Columbia University. He is editor of four books and over eighty articles in the areas of learning computer systems, reading comprehension, and memory. He is a consultant to IBM, Citicorp, ITT, and Xerox.

SETH CHAIKLIN is a senior research associate at the Institute for Learning Technologies, Teachers College, Columbia University. His current interests are the role of telecommunications in society, physics education research, and the relation between theory and educational practice.

NANCY CUNNIFF is a doctoral candidate at Teachers College, Columbia University, studying computing in education. She has taught programming for several years to students of all ages. Her major research interest is the investigation of how graphical representation of conceptual materials affects learning.

BEN DAVIS is the director of the Visual Courseware Group at Project Athena at the Massachusetts Institute of Technology and is the chair of the research consortium for the National Demonstration Laboratory for Interactive Technology at the Smithsonian Institution.

GEORGE FRANZ teaches at The Computer School, New York City Board of Education. He is a consultant to the National Science Resources Center/Smithsonian Institution, has taught at Bank Street College of Education, and was formerly the computer editor of a major parents' magazine. He is also executive director of the World Economic Organization in New York City, and has recently delivered a speech, "Alternative Exchange System for World Stability," at the United Nations.

MATTHEW W. LEWIS, initially trained as a teacher, is currently a research associate at Carnegie Mellon University, studying the acquisition of complex cognitive skills and the application of artificial intelligence techniques to teaching.

ROBERT O. McCLINTOCK chairs the department of communication, computing, and technology in education and also directs the Institute for Learning Technologies at Teachers College, Columbia University. A cultural historian, McClintock seeks to improve the study of our humanistic heritage through creative use of information technology.

DON NIX is manager of exploratory education systems at IBM's T. J. Watson Center, Yorktown Heights, New York. His main interests are in humanistic computing, where technology is useful to people as individuals rather than as members of organizations. Typical domains are reading comprehension, children and creativity, and computer languages.

SEYMOUR PAPERT is Cecil Greene Professor of Education, Massachusetts Institute of Technology.

CARLA SEAL-WANNER is in the department of communications, computing, and technology in education at Teachers College, Columbia University.

DANIEL L. SCHWARTZ is a student in the department of communications, computing, and technology at Teachers College, Columbia University. His interests include applying cognitive science to the development of instructional tools.

KAREN SWAN is director of computer science education at the Bicultural Day School, Stamford, Connecticut. She is consultant to the Office of Educational Assessment, New York City Board of Education, and research assistant, Institute for Learning Technologies, Teachers College, Columbia University.

ROBERT P. TAYLOR is associate professor of computing and education and director of the Center for Intelligent Tools in Education at Teachers College, Columbia University. He has published and consulted widely on computing and education since 1975.

Index